1. AGENC. (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	12/93	Final 1/15/93 - 12/31/93

4. TITLE AND SUBTITLE

The Use of Fuzzy Set Classification for Pattern Recognition of the Polygraph

5. FUNDING NUMBERS

N00014-93-1-0570

(1)

6. AUTHOR(S)

Benjamin Knapp, Mitra Dastmalchi, Eric Jacobs, Shahab Layeghi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

San Jose State University
San Jose, CA 95192-0084

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
Arlington, VA 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified unlimited

This document has been approved for public release and sale; its distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

See abstract on face page

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE.	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
U	U	U	Unlimited

Pattern Recognition of the Polygraph Using Fuzzy Classification

Shahab Layeghi, Mitra Dastmalchi, Eric Jacobs, and R. Benjamin Knapp
Department of Electrical Engineering
San Jose State University, San Jose, California 95192-0084

Abstract- Polygraph tests are a widely used method to distinguish between truth and deception. Polygraph charts are usually analyzed by human interpreters. However, computer algorithms are now being developed to score the tests or verify the results. These methods are based on statistical classification techniques. In this study a number of time, frequency and correlation domain features were selected and used. The fuzzy K-nearest neighbor algorithm was used to classify the polygraph charts, a correct classification of ninety-one percent was obtained for a set of one hundred case files supplied by the NSA.

I. Introduction

Polygraph examinations are the most widely used method to distinguish between truth and deception. In a polygraph examination a person is connected to a special instrument called a Polygraph which records several physiological signals such as electrocardiogram, galvanic skin response, and respiration. During the polygraph examination, the subject is asked a set of questions by an examiner. The examiner analyzes the graphs to determine the reactions of the subject to the questions for evidence of truth or deception.

Different formats are used for polygraph examinations. A given polygraph test format is an ordered combination of relevant, irrelevant and control questions. Relevant questions are questions about a specific issue. Control questions are not directly related to the specific issue under question but are designed to make the subject uncomfortable and provide a physical response for comparison. Irrelevant questions are very general questions that are not related to the issue but provide a response to comparison [1][4]. The rational for scoring the tests is that a deceptive subject will be more threatened by the relevant questions than by the control questions while a non deceptive subject will be more threatened by the control questions than the relevant questions.

Three general types of test formats are in use today. These are Control Question Tests, Relevant-Irrelevant Tests, and Concealed Knowledge Tests. Each of the general test formats may have a number of more specific variations. Each test consists of two to five charts containing a prescribed series of questions. The test format that is used in an examination is determined by the test objective [3][4].

A control question test is often used in criminal investigations. The control questions are compared to the relevant questions and if the responses to the relevant questions are greater, the subject is usually classified as deceptive. Irrelevant questions are used as buffers.

The problem with human classification of polygraph tests is that the outcome depends on the examiner's experience and judgment. As a result, automatic scoring systems to classify polygraph tests are being developed to overcome this problem. Several methods for polygraph classification have been studied which are mostly based on statistical classification techniques [1] [2]. This project, however, is focused on using fuzzy classification rather than statistical methods.

II. Methods

Digitized polygraph data used in this project were collected from various police stations by the National Security Agency (NSA). The data files were organized according to the test format used and were decoded to ASCII format so they can be read by the mathematical computation package, MATLAB. All preprocessing and feature extraction routines were implemented in MATLAB.

Classification of polygraph charts like any other pattern recognition problem can be divided into two major sections, feature extraction and classification. The methods used for each one of these sections are explained in the following section.

A. Feature Extraction

Polygraph data consists of signals from four different channels: Galvanic Skin Response (GSR), electrocardiogram, higher respiration, and lower respiration. Before actual feature extraction was done, the data was preprocessed. The electrocardiogram signal was decomposed into a high frequency component showing heart pulse, and a low frequency component showing blood volume. The derivative of the blood volume was also used as a preprocessed channel. In order to eliminate any noise and trend, these six derived signals were detrended and filtered.

A broad range of features that are the best indicators of truth or deception were chosen based on previous work and on interviewing polygraph examiners[5][6]. In general, features are divided into three main groups, time-domain features, frequency-domain features and correlation features. Time-domain features involved standard statistical characteristics such as the mean, the standard deviation, and the median for each of the six channels. Other channel-specific time-domain features such as the ratio of inhalation over exhalation and the auto-regressive parameters of a tenth order AR filter model for the heart pulse were also considered as features. Frequency-domain features for each of the six channels included the fundamental frequency, the magnitude of the power spectral density at the fundamental frequency, and the coherency at the fundamental frequency. To extract each feature for each question a time fragment of each signal was selected starting several seconds after a question was asked and continuing for a number of seconds. The exact time frame used was dependent on the channel being measured.

A total of ninety-nine different features were extracted for each question in each chart. Each feature was extracted for each relevant, irrelevant, and control question in the test. In order to classify subjects using the difference between the control and relevant responses similar features for these question were combined. Seven methods were used to then combine each control and relevent features into one common feature. (The irrelevant features were not used.) The first method was subtracting the average response normalized to the control question from average response normalized to the relevant question. The second method was to use maximum response in place of average response. The other methods of combining control and relevant features were max - min, min - max, min - min, dividing the averages, and using normalized averages.

B. Classification

It was decided to use the K-nearest neighbor (KNN) classifier in this project because the distribution of the samples of deceptive and non-deceptive classes were not known beforehand, and the KNN classifier does not explicitly use the distribution of the samples.

One of the characteristics of the conventional KNN classification method is that it assigns each input to one of the possible classes (crisp classification). The way that humans think and classify objects is fundamentally different. Each object can be considered to belong to more than one class at the same time, and there are degrees of membership for each class. This is the basic idea that is followed in fuzzy logic. It was decided to use a modified version of KNN algorithm which uses fuzzy logic concepts [7] [8]. In this way the output will be the *possibility* of deception and thus give a continuous measure of truth versus deception rather than a discrete choice.

The first step in the fuzzy KNN algorithm is the same as first step in crisp classifier. In both cases K nearest neighbors of the input are found. In the crisp classifier, the majority class of the neighbors is used to assign the input to a class. In the fuzzy classifier, the membership of the input to each class is found. In order to do so, the membership vector of each neighboring sample is combined to obtain the membership vector of the input. If the samples are crisply classified, membership vectors should be assigned to them. One method to do so is to assign the membership of 1 to the class that it belongs to, and membership of 0 to other classes. Other methods assign different memberships to the samples according to their distance from the mean of the class, or the distances from the nearby samples of its own class and the other classes.

When the membership vectors of the labeled samples are specified, they are combined to find the membership vector of the unknown class. This procedure is done in a way that samples that are closer to the input have more effect on the resultant membership function. The following formula uses the inverse distance to weigh the membership functions. x is the input to be classified, x_j is the j*th* nearest neighbor and u_{ij} is the membership of the j*th* nearest neighbor of the input in class i. D(x,y) is a distance measure between the vectors x and y. Euclidean

distance has been used as the distance measure in this project.

$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}(1/D(x,x_j)^{\frac{1}{m-1}})}{\sum_{j=1}^{K}(1/D(x,x_j)^{\frac{1}{m-1}})}$$

where m is a parameter that changes the weighing effect of the distance.

When $m >> 1$, all the samples will have the same weight. When $m \rightarrow 1$, nearest samples have much more effect on the membership value of the input [7].

The feature extraction mentioned in section II.A created 669 features for each chart. This number of features was larger than could be practically used by fuzzy KNN classifier. It was decided to reduce the number of features to 30 at this step. Two different methods were chosen to test the features one at time to find the best 30 features. The first method was using the fuzzy KNN classifier to classify the data files using one feature at a time. The classifier parameters such as K and threshold were changed to find the best classification results. The value 5 was selected for K because it gave better classification results. Also, a threshold of 0.5 was used to defuzzify the output of the classifier. The second method was using the scatter criterion presented below:

$$J = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \qquad (1)$$

where m_i is the mean of the class i, and s_i is the standard deviation of class i.

This criterion measures the distance between the means of the two classes, normalized over the sum of the variances. Therefore, the more compactly the samples in each class are separated, the higher will the value of J.

The results of KNN and scatter criterion were averaged for three sets of data. Thirty features that showed the best performance in both methods or had a special significance to the polygraph examiner were selected.

Better classification was achieved by combining several features. The most basic way of finding the best combination is the exhaustive search method. That is trying all the combinations for these features. This is not practical when the number of features is large.

All combinations of two features out of the best 30 features were tried. Then, the 20 combinations with the best accuracy rate were selected and combined with other features of the best 30 set to build combinations of three. The same procedure was followed for combinations of three and four with the best selected set of features. This procedure was continued until adding features did not improve the classification results significantly.

III. Results

The classification results were improved by increasing the number of combined features from 1 to 4. Using single features the best result was 70 percent by using the mean of GSR signal. When combinations of two features were used the best result was obtained using the difference between maximum and minimum of the GSR and the maximum of derivative of the low cardio signal. The average result was 73 percent. By combining three features the best result was 78 percent by using the maximum of the GSR, the maximum of the upper respiratory, and the frequency of the maximum integrated spectral difference of the control-relevant pair in the GSR. The combinations of four features that showed the best classification results are shown in Table 1. These results were obtained by using K=5 and defuzzification threshold of 0.5 in the fuzzy K-nearest neighbor classifier. The feature set that showed the best result on the average was used for further experiments. These features were the maxima of the GSR, the high cardio and the upper respiratory signals and the difference between the maximum and the minimum of the high cardio signals.

Different values for K and the defuzzification constant were tried to optimize the classifier. The best result was obtained using K = 6 and the defuzzification constant of 0.6. The average result for three sets was 81.6 percent.

Another experiment that was performed was combining the results of several charts that are used in a polygraph test. Usually a polygraph test is composed of two to four charts that contain the same questions. Previously, the charts were classified independently. The outcome of classifying every chart in a test was added and the whole test was classified accordingly. Correct classification results for sets one to three were 85.7, 80.0, and 91.4 percent.

IV. Conclusion and Discussion

Set	Features				Accuracy
Set 1	GSR(max)	HC(max-min)	LR(max)	UR(max)	81.0
	GSR(max)	HC(min)	LR(max)	UR(max)	80.2
	GSR(max)	LR(max)	UR(max)	GSR(isd)	74.4
Set 2	GSR(max)	DLC(mean)	UR(max)	GSR(isd)	81.0
	GSR(max)	HC(min)	LR(max)	UR(max)	79.4
	GSR(max)	LR(max)	UR(max)	GSR(isd)	79.0
Set 3	HC(max-min)	DLC(mean)	UR(max)	GSR(isd)	87.4
	GSR(max)	LR(max)	UR(max)	GSR(isd)	86.6
	GSR(max)	HC(max-min)	LR(max)	UR(max)	82.5
Average	GSR(max)	HC(max-min)	LR(max)	UR(max)	81.0
	GSR(max)	LR(max)	UR(max)	GSR(isd)	80.0
	GSR(max)	HC(min)	LR(max)	UR(max)	79.8

GSR=Galvainc Skin Response, HC=High Cardio, LC=Low Cardio, DLC=Derivative of Low Cardio, UR=Upper Respiratory, LR=Lower respiratory isd=integrated spectral density.

Table 1. classification results with combining 4 features

The classification results improved consistently by increasing the number of features in the combination from one to four. The feature set that showed the best classification result only included simple time-domain features which were the maximums of GSR, lower respiratory, upper respiratory, and the difference between maximum and minimum of the high cardio signal. It is notable that these features come from different channels. It is possible to conclude that adding the information in different physiological channels is a good way of finding the evidence of deception in polygraph tests. Although these features showed the best results, some other features appeared in other combinations with approximately the same results. In future work combinations of more than four features could be studied in order to find an optimum feature set that uses all possible features.

The primary advantage of using the fuzzy KNN classifier is that it gives the possibility of deception rather than just classifying the person as deceptive or non-deceptive. This gives the examiner the ability to have a continuous measure of deception. Also using a fuzzy classifier whose membership functions could be trained during a polygraph exam may direct the examiner toward a specific line of questioning.

[This work was supported by a grant from the National Security Agency.]

References

[1] Dale E. Olsen, et. al., "Recent developments in polygraph testing: A research review and evaluation - A technical memorandum, " Washington DC: US Government Printing Office 1983.

[2] John C. Kircher and David C. Raskin, "Human versus computerized evaluations of polygraph data in a laboratory setting, " Journal of Applied Psychology, Vol.73, 1988 No 2, pp. 291-308.

[3] John E. Reid and Fred E. Inbau, "Truth and Deception: The Polygraph (Lie Detector) Technique", The Williams & Wilkins Company, Baltimore, Md., 1966

[4] Michael H. Capps and Norman Ansley, "Numerical Scoring of Polygraph Charts: What Examiners Really Do", Polygraph, 1992, 21, 264-320

[5] Brian M. Duston, " Statistical Techniques for Classifying Polygraph Data ", Naval Command Control and Ocean Surveillance Center, RDT&E Division

[6] Howard W. Timm, " Analyzing Deception From Respiration Patterns " , Journal of Police Science and Administration, 1982, 1, 47 - 51.

[7] J.M. Keller, M.R. Gray and J.A. Givens, "A Fuzzy K Nearest Neighbor Algorithm", IEEE Trans. on Syst. Man. Cybernetics, vol SMC-15, no. 4 (1989)

[8] J.C. Bezdek and Siew K. Chuah, "Generalized K-Nearest Neighbor Rules, Fuzzy Sets and Systems vol. 18 (1986)

Appendix B
Progress Report

1. Overview

A. Development of Data Parsing Algorithm

The first phase of this project was to be able to read the MGQT data files received from the NSA and separate this data into appropriate features for classification. After consulting with the University of Washington, we were able to develop our own data reading program.

After consultation with experienced polygraph examiners and a detailed review of the polygraph literature, the data reading program was then modified to parse the data into a matrix of features. The feature set included, as outlined in the project proposal, time domain, frequency domain, and correlation domain data. Some examples of the feature set are:

Time Domain Features

- Mean, curvelength, area, and standard deviation for all polygraph channels
- Average of the amplitudes of the peaks in the cardio and respiratory channels
- Derivative of the amplitudes of the peaks of cardio and respiratory channels
- Number of peaks in the cardio and respiratory channels
- Inhalation amplitude/exhalation amplitude of respiratory channels

Frequency Domain Features

-Fundamental frequency of cardio and respiratory signals
-Coherancy and cross power spectral density between cardio and respiratory channels
-Power spectral density of cardio and respiratory channels
-Integrated power spectral density for cardio channel

Correlation Domain Features

- Autoregressive parameters (10) for cardio signal
- Cross-correlation between cardio and respiratory channels

B. Design of Fuzzy Classifier Algorithm

Fuzzy classifier design has focused on the development of a fuzzy set based *k nearest neighbor* algorithm. The algorithm learns using a set of MGQT data divided equally between truthful and deceptive. Since there were 150 deceptive files and only 50 truthful files, the deceptive files were divided into three sets of 50 files each. The algorithm was trained separately for each data set. When a question was asked more than once by an examiner the questions were

scored individually and then combined at the end on a majority basis. Some examples of the results achieved using the best four features and no indecision allowed are:

Deceptive Set	%Correct Deceptive	%Correct Truthful	%Correct Total
1	94	78	86
2	89	72	80
3	100	83	91

The following are three reports which describe in detail the work performed. In addition, a copy of a paper which has been submitted to the IEEE International Conference on Fuzzy Systems is also included. Finally, a manual is included which instructs the user how to repeat the work performed at SJSU.

Features Analysis of the Polygraph

A Report
Presented to
The Faculty of the Department of Electrical Engineering
San Jose State University

In Partial Fulfillment
of the Requirements for the degree
of Master of Science

By
Mitra Dastmalchi
December 1993

Acknowledgement

I express may sincere appreciation to all of those who have contributed to this project. Special recognition goes to my advisor, Dr. Ben Knapp, for his advise and encouragement. I am also grateful for the help of my partners, Shahab Layeghi and Eric Jacobs. Especially shahab, for his support and valuable suggestions.

Feature Analysis of the polygraph

By
Mitra Dastmalchi

Sponsor: Dr. Benjamin Knapp

Approved: _____ _____
Sponsors Signature Date

Graduate Commitee

	Name	Date
Dr. Sun Chiao	_____	_____
Dr. Richard Duda	_____	_____
Dr. Peter Reischl	_____	_____
Dr. Avtar Singh	_____	_____

Graduate Coordinator

Dr. Rangaiya Rao	_____	_____

Mitra Dastmalchi, 25800 Industrial Blvd Hayward, CA 94545 Tel: (510)782-3104

Acknowledgement

I express may sincere appreciation to all of those who have contributed to this project. Special recognition goes to my advisor, Dr. Ben Knapp, for his advise and encouragement. I am also grateful for the help of my partner, Shahab Layeghi and Eric Jacobs. Especially shahab, for his support and valuable suggestions.

Contents

0 Introduction

1 Polygraph

1.1 Polygraph Examination
1.2 History
1.3 Modern Test Format
1.4 Present Day Equipment

2 Classifier Algorithm

2.1 K-Nearest Neighbor Algorithm

3 Frequency and Correlation Domain Features

3.1 Preview
3.2 Fundamental Frequency
3.3 AR Modeling
3.4 Cross Correlation Function
3.5 Whitening Filter
3.6 Spectral Analysis
3.7 Integrated Spectral Difference

4 Feature Extraction

4.1 Preprocessing
4.2 Feature Selection

5 Results

5.1 Discussion
5.2 Frequency Domain Features Clustering

Conclusion

Appendix A

Appendix B

0 Introduction

The polygraph examination is one of the most popular methods to measure deception. Polygraph tests are used in criminal investigations to determine if a suspect is being deceptive when answering the questions concerning a crime. During a polygraph test, the subject is asked a series of control, relevant and irrelevant questions that provide physiological responses for comparison with question that are relevant to the investigation. The three physiological responses that are currently measured are electrocardiogram, galvanic skin response and respiration. The controversy surrounding the use of polygraph tests centers on the subjective judgment of polygraph examiners in classifying the subject as deceptive or non-deceptive. The object of this project is to develop an automatic scoring system to overcome this perception. The computer algorithm will be able to use more sophisticated techniques than human examiners, should be more accurate and will ensure consistency from case to case.

In order to implement the automatic scoring system, two main algorithms were developed. These were: the feature extraction algorithm, which process the polygraph data in three time, correlation and frequency domains, and the fuzzy classifier algorithm, which accepts the features and determines the possibility of deception. Because of the nature of the input, fuzzy logic was chosen to implement the system which gives the possibility of belonging of an input to each class. Initially, a set of features based on physiological reactions were selected. Then, the fuzzy K-nearest neighbor classifier was used to classify the features.

1 Polygraph

1.1 Polygraph Examination

The primary use of the polygraph test is during the investigation stage of the criminal justice process. In addition to the significance role in criminal justice, they are also used for national security, intelligence and counterintelligence activities [1]. The three physiological responses currently obtained from a polygraph examination are electrocardiogram, respiration and galvanic skin response. Electrocardiogram is measured by placing a standard cuff on the arm over the brachial artery. Respiration is monitored by placing rubber tubes around the abdominal area of the subject. Skin conductivity is measured by electrodes placed on two fingers of the same hand of the subject [1].

The effectiveness of a polygraph examination is often the result of the test format that is used. A polygraph test format is an ordered combination of relevant question about an issue, control questions that provide physiological responses for comparison and irrelevant questions that act as a buffer [1]. An example or a relevant question is, " did you embezzle any of the missing $12000?" The corresponding control question would be about stealing; an example is, "did you ever steal money or property from an employer?" The example of an irrelevant question is, " is your name John?" Irrelevant questions are answered truthfully and are not stressful. The rational for scoring these tests is that a deceptive subject will be more threatened by the relevant question than by the control question while a non deceptive subject will be more threatened by the control questions than the relevant question.

Polygraph charts are usually analyzed by a human interpreter for evidence of truth or deception. A control question polygraph chart usually consists of 3 sets of control relevant question pairs separated by neutral questions. The examiner scores the charts by comparing each relevant question. For each of three physiological responses, he will give a numerical score ranging from -3 t0 +3, depending on the magnitude of the difference. He then adds up scores for all control relevant pairs. If the score is below threshold value, he scores the chart as deceptive or non deceptive.

Sometimes the examiner can not make a clear decision and must score the chart as inconclusive. The examiner's decision will be based on his or her experience and training. For example, a change in the polygraph tracing considered by one examiner as a physiological changes, may be considered by another as an artifact of the recording system. In an effort to eliminate the inconsistencies involved in interpreting polygraph data, computer algorithm are being developed.

1.2 History[1]

The first attempt to use a scientific instrument in an effort to detect deception occurred around 1895 [2]. That was the year that Cesar Lombroso published the results of his experiments in which a hydrosphygmograph was used to measure the blood pressure-pulse changes of criminals in order to determine whether or not they were deceptive. Although the hydrosphygmograph was originally intended to be used for medical purposes, Lombroso found that it worked well for lie detection. Lombroso may have been the first to use a peak of tension test format. This was done by showing a suspect a series of photographs of children, one being the victim of sexual assault. If the suspect did not react more to the victims picture than the pictures of the other children, Lombroso concluded that the suspect did not know what the victim looked like and therefore was not the alleged perpetrator.

In 1914 Vittorio Benussi published his research on predicting deception by measuring recorded respiration tracings [3]. He found that if the length of inspiration were divide by the length of expiration, the ratio would be larger after lying than before lying and also before telling the truth than after telling the truth. In 1921 John A. Larson constructed an instrument capable of simultaneously recording blood pressure pulse and respiration during an examination [2][3]. Larson reported accurate results which prompted Leonarde Keeler to construct a better version of this instrument in 1926 [2][3].

The use of galvanic skin response in lie detection began during the turn of the century. It's usefulness, however, did not become evident until the 1930's during which time several articles written by Father Walter G. Summers of Fordham University in New York [3]. In these articles he reports over 90 criminal cases in which examination using the galvanic skin response had all been successful and confirmed by confession or supplementary evidence. The usefulness of the galvanic skin response prompted Keeler to add an galvanometer to his polygraph. At the time of Keelers death in 1949, the Keeler Polygraph recorded blood pressure-pulse, respiration, and galvanic skin response [3].

1.3 Modern Test Formats[1]

The effectiveness of a polygraph examination is often the result of the test format that is used. A polygraph test format consists of an ordered combination of relevant questions about an issue, control questions that provide a physical response for comparison, and irrelevant questions that also provide a response or the lack of a response for comparison [1][3]. Three general types of test formats are in use today. These are Control Question Tests, Relevant-Irrelevant Tests, and Concealed Knowledge Tests. Each of the general test formats may have a number of more specific variations. Each test consists of two to

[1]These sections were exerpted from Jacobs [10].

five charts containing a prescribed series of questions. The test format that is used in an examination is determined by the test objective [2][3].

The concealed knowledge test, also called peak of tension test, is used when facts about a crime are known only by the investigators and not by the public. In this case, a subject would not know the facts unless he or she was guilty of the crime. For example, if a gun was used in a crime and the public did not know the caliber, an examiner could ask a suspect if it was a 22 caliber, a 38 caliber, or a 9 mm. If the gun used was a 9 mm and the suspect was deceptive, a polygraph chart would probably indicate evidence of deception.

A control question test is often used in criminal investigations. Relevant-Irrelevant tests are usually used to test people trying to obtain security clearance or get a job. In this test, relevant questions are compared to irrelevant questions. Very few control questions are asked. The purpose of control questions in this test is to make sure that the subject is capable of reacting at all.

1.4 Present Day Equipment[2]

The most popular polygraph machines today are the Reid Polygraph developed in 1945 and the Axciton Systems computerized polygraph developed in 1989 [1][4]. The Reid polygraph scrolls a piece of paper under pens that record the biological signals. The Axciton polygraph digitizes physiological signals and uses a computer to process them. The sampling frequency of the Axciton machine is 30 Hz. Axciton provides a computer based system for ranking the subject responses but allows printouts of the charts to be scored by hand the traditional way.

Both machines record the same biological signals using standard methods. Blood pressure is measured by placing a standard blood pressure cuff on the arm over the brachial artery. Respiration is monitored by placing rubber tubes around the abdominal area and the chest of the subject. This results in two signals, an upper and lower respiratory signal. Skin conductivity is measured by placing electrodes on two fingers of the same hand.

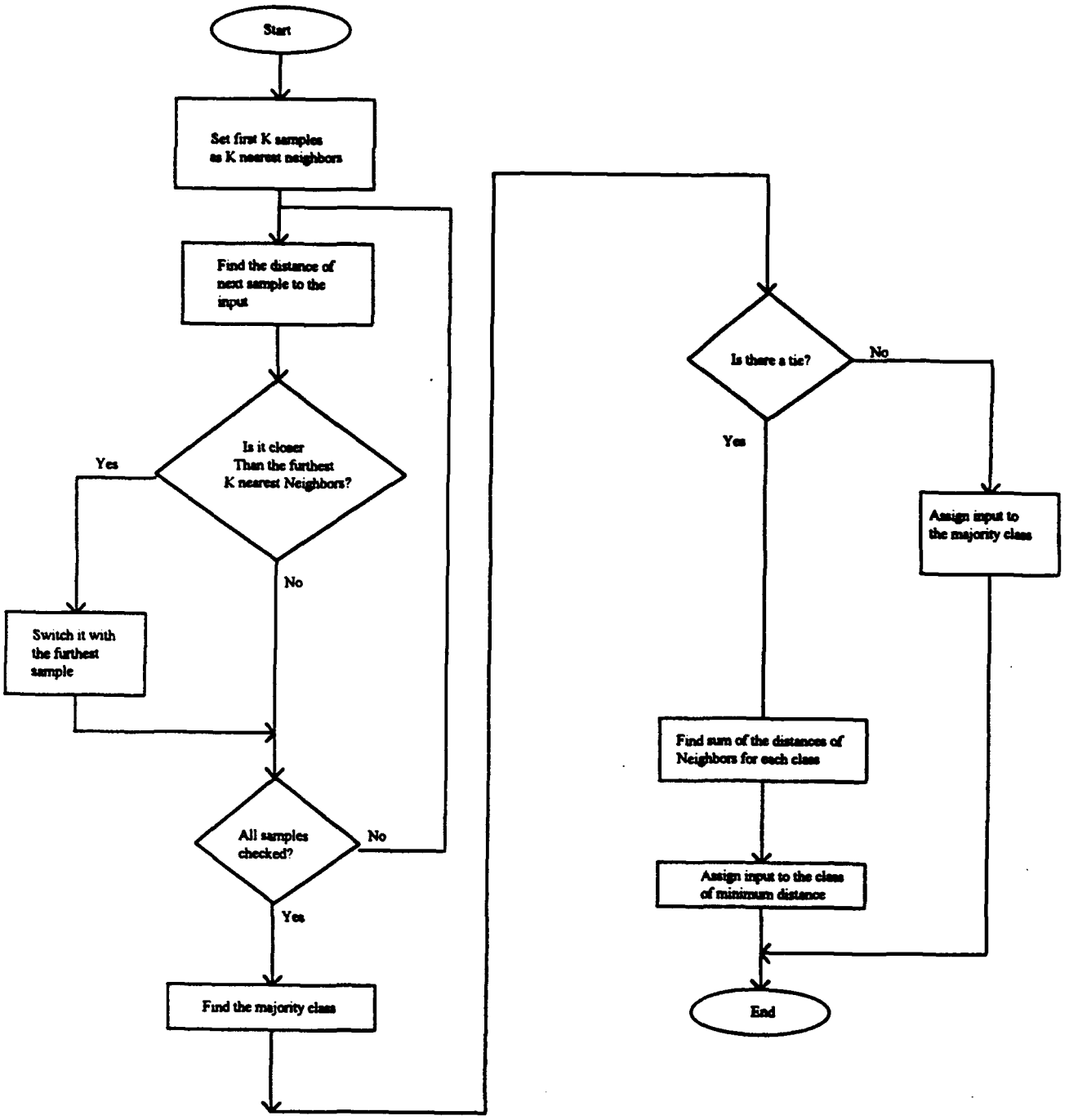
[2]This section was exerpted from Jacobs [10].

2 Classifier Algorithm

2.1 K-Nearest Neighbor Algorithm[3]

K-nearest neighbor algorithm is a supervised classification method. There is no need for the training or adjusting the classifier. A set of labeled input samples is given to the classifier. When a new sample is given to the system, it finds its K nearest neighboring samples, and assigns this sample to the class that the majority of the neighbors belong to. K could be any positive integer. When K is set to 1, the algorithm is called the nearest neighbor algorithm. In this case each new sample is assigned to the class of its nearest neighbor. If K is greater than 1, it is possible that there is no majority class. To remove this tie, the sum of the distances of the new sample to its neighbors in each class is computed and the sample is assigned to the class that has the minimum distance. The main advantage of using this method is that the samples of each class are not needed to cluster in a pre specified shape. For example, for a two class classification, the K-nearest neighbor classifier can still give very good results if the samples of each class are clustered in two distinct points in the space. The algorithm for the K nearest neighbor is shown in flow chart 1. It is supposed that C is the number of classes, K is the number of neighbors in KNN, $x_i x_i$ is the *ith* labeled sample and y is the input to be classified.

[3]This section was exerpted from Layeghi [11].



Flow chart 1. Fuzzy K Nearest Neighbor Algorithm

The fuzzy K nearest neighbor algorithm uses the same idea of conventional K nearest neighbor algorithm, that is finding the K samples that are closest to sample to be classified. But there is a conceptual difference in classification. When fuzzy classification is used, the input is not assigned to a single class. Instead, the degree of belongings of the input to each class is determined by the classifier. By using this method more information is obtained about the input. For example if the result of classification determines membership of an input to class A is 0.9 and to class B is 0.1, it means the input belongs to class A with a very good possibility. But if the membership to class A is 0.55 and to class B is 0.45, it means that we cannot be very sure about the classification of the input. If the crisp classifier is used, in both cases the input will be assigned to class A and no further information is obtained.

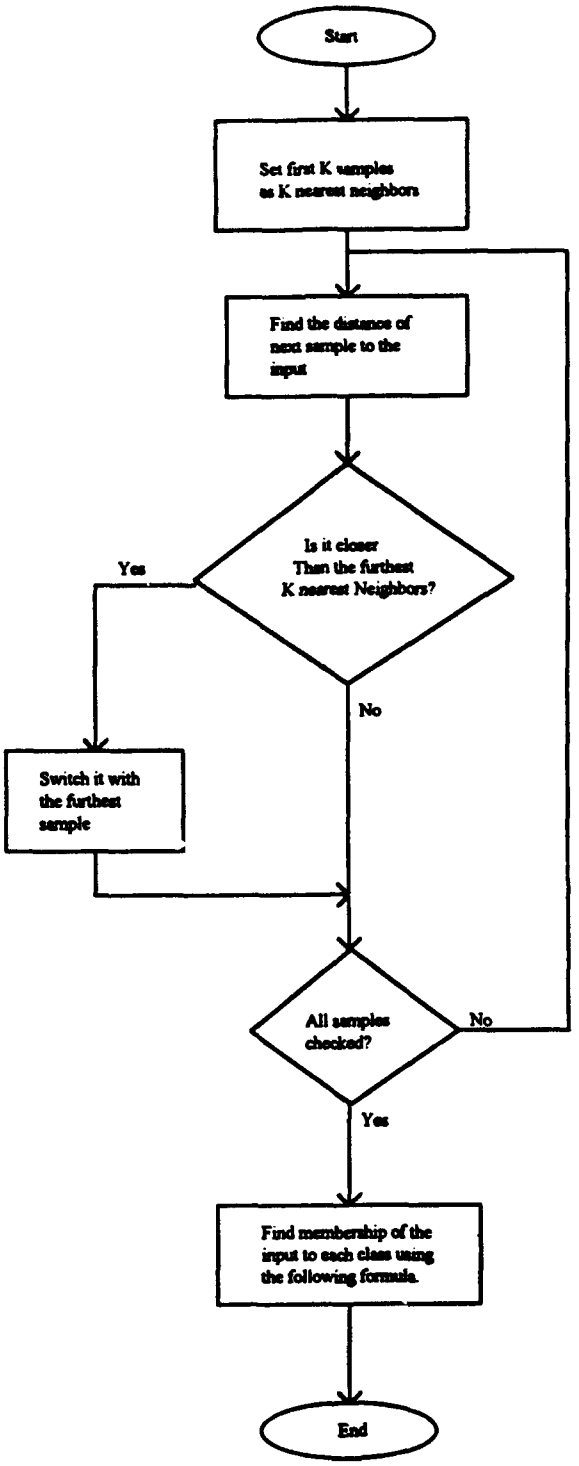
Refer to [5] [6] for more detailed discussions about fuzzy K nearest neighbor algorithms. The flowchart for a fuzzy K nearest neighbor classifier is drawn in flow chart 2.

The first step in the fuzzy K nearest neighbor algorithm is the same as first step in crisp classifier. In both cases K nearest neighbors of the input are found. While in crisp classifier the majority class of the neighbors is assigned to the input, in Fuzzy classifier membership of the input to each class should be found. In order to do so the membership vector of each sample is combined to obtain the membership vector of the input. If the samples are crisply classified, membership vectors should be assigned to them. One method to do so is to assign the membership of 1 to the class that it belongs to, and membership of 0 to other classes. Other methods assign different memberships to the samples according to its distance from the mean of the class, or the distances from the nearby samples of its own class and the other classes.

When the membership vectors of the labeled samples are specified, they are combined to find the membership vector of the unknown class. This procedure should be done in a way that samples that are closer to the input have more effect on the resultant membership function. The following formula uses the inverse distance to weigh the membership functions. x is the input to be classified, x_j is the j*th* nearest neighbor and u_{ij} is the membership of the j*th* nearest neighbor of the input in class i. D(x,y) is a distance measure between the vectors x and y which could be the Euclidean distance.

$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}\left(1/D(x,x_j)^{\frac{1}{m-1}}\right)}{\sum_{j=1}^{K}\left(1/D(x,x_j)^{\frac{1}{m-1}}\right)}$$

m is a parameter that changes the weighing effect of the distance. When m >> 1, all the samples will have the same weight. When m approaches 1, nearest samples have much more effect on the membership value of the input.



Flow chart 2. Fuzzy k nearest neighbor

$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}(1/D(x,x_j)^{\frac{1}{m-1}})}{\sum_{j=1}^{K}(1/D(x,x_j)^{\frac{1}{m-1}})}$$

3 Frequency and correlation Domain Features

3.1 Preview

The purpose of this chapter is to show how the frequency and correlation domain representations of polygraph signals can be used effectively in polygraph analysis. The first step in analysis of a time series is to plot the data and to obtain simple descriptive measures of the main properties of the series. For some series, in addition to features such as trend, seasonal effect and cyclic changes, more sophisticated features such as mean, variance, auto correlation and frequency content will be required to provide an adequate analysis.

Most physical processes, including polygraph signals, involve a random element in their structures. Currently, human examiners score polygraph tests by analyzing obvious features in the time domain. It is presumed that processing polygraph signals in frequency and correlation domain will provide features which are discriminator between deceptive and non-deceptive subjects. Before finding the frequency domain features the trend in the electrocardiogram channel was eliminated. In order to do so, a high frequency electrocardiogram channel, called heart pulse, is produced by highpass filtering it.

The goal of this chapter is to explain the techniques used to extract appropriate features in frequency and correlation domains. The methods for estimating features of the polygraph signals such as fundamental frequency, spectral density and cross correlation between the channels will be discussed.

3.2 Fundamental Frequency

One feature which is considered important in the frequency domain is the fundamental frequency of the signal. The purpose of finding the fundamental frequency is to classify the way the frequency changes in a specific time segment. The assumption in polygraph signals is that the frequency of the signal changes after a relevant or a control question is asked. Different methods have been proposed to find the fundamental frequency of a signal. One of these methods is using the auto correlation function.

The auto correlation representation of a signal is a convenient way of displaying certain properties of the signal. For example, the auto correlation function of a periodic signal is also periodic with the same period. For periodic signals with period P, the auto correlation function attains a maximum at samples $0, \pm P, \pm 2P, \ldots$. Regardless of the time origin of the signal, the period can be estimated by finding the location of the first maximum in the auto correlation function [7].

This property makes the auto correlation function an attractive basis for estimating periodicity in most signals including the electrocardiogram and respiration signals of the polygraph records. Therefore, a short segment of the signals (electrocardiogram and respiratory) after each question is selected and pre-processed. The auto correlation is then calculated for the windowed segments of the heart pulse and respiratory signals using MATLAB. Figure 1 shows the examples of auto correlation functions computed for heart pulse with N = 150 and upper respiratory with N = 400 sampled at 30 Hz. N is the number of samples .

It is noticeable that the auto correlation functions of the above signals are a mixture of damped exponential and sinusoids. For the heart pulse, peaks occur approximately at multiples of 20 samples indicating a period of 20/30=0.67 seconds or a fundamental frequency of approximately 1.5 Hz. For the upper respiratory, peaks occur approximately at multiples of 133 samples indicating a period of 133/30 = 4.4 seconds or a fundamental frequency of approximately 0.23 Hz.

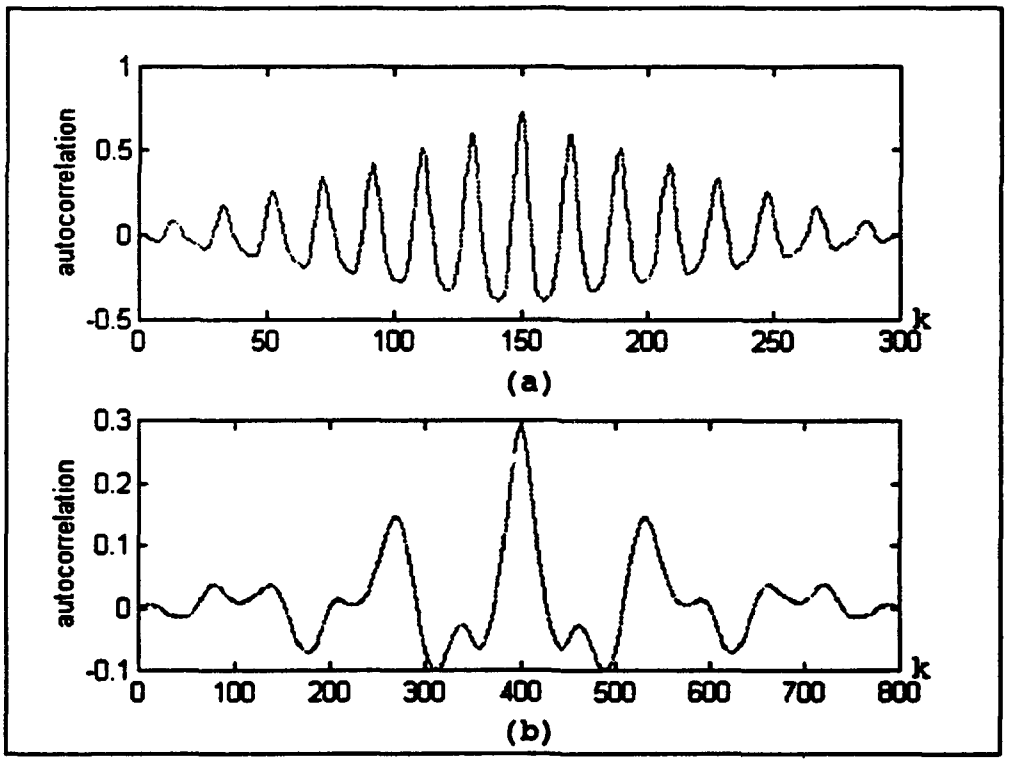


Figure 1. Plots of auto correlation function for (a) heart pulse and (b) upper respiratory where k is the number of samples.

For some subjects, the period of the electrocardiogram or upper respiratory signal changes across the N sample interval. Also, the shape of the signal varies somewhat from period to period. Because of the finite length of segments involved in the computation of auto-correlation, there is less and less data involved in the computation as the lag increases. This leads to the reduction in amplitude of the correlation peaks as lag increases.

An important issue is how N should be chosen to give a good indication of periodicity. Because we are interested in observing changes in signal after the question is asked, N should be small. On the other hand, it should be noted that to get any indication of periodicity in the auto correlation function, the window must have the duration of at least two periods of the waveform. In order to choose the best N, the fundamental frequency for different time frames without overlap were calculated and the results were examined. The fundamental frequencies of heart pulse for the four second frame are shown in Table 1 and 2 in Appendix A. No single value of N is entirely satisfactory because the frequency changes from individual to individual. However, a suitable practical choice for N was chosen on the order of 180 and 480 for heart pulse and upper respiratory respectively.

3.3 Modeling

Detailed information about a time series can be obtained from creating a model. In this section a model will be found for the heart pulse signal. Finding a suitable model for a given time series depends on the properties of the series and the number of observations available. In signal modeling the output signal is known and the model development is based upon the fact that signal points are correlated. Estimated auto correlation function (ACF) of the time series is helpful in identifying which type of ARMA model is appropriate and gives the best representation of the signal.

The ACF of a MA process cuts off at lag q whereas the ACF of an AR process is a mixture of damped exponential and sinusoids and dies out slowly. For example, if r1 is significantly different from zero but the subsequent values of r_k are all close to zero then an MA(1) model is indicated since its theoretical ACF is of this form. Alternatively, if r_1, r_2, r_3 ,.. appear to be decreasing exponentially, then an AR(1) model may be appropriate.

It is usually difficult to find the order of an AR process from the sample ACF alone. A model with too low an order will not represent the properties of the signal. Also a model with too high an order will represent any measurement noise or inaccuracies. Therefore, neither a high order nor a low order model will be a reliable representation of the signal. As a result, method that will determine the model order should be used. One approach is to fit AR processes of progressively higher order, to calculate the squared error for each value of model order (M), and to plot this against model order. It may then be possible to see the value of M where the curve flattens out and the addition of extra parameters gives

little improvement in fit. Another approach based upon the principals of prediction is that to increase the model order until the residual process becomes a white noise.

Other criteria have been developed that are based upon concepts in mathematical statistics [9]. The first one is the final prediction error (FPE),

$$\text{FPE} = P\frac{N+M+1}{N-M-1} \qquad (3.3a)$$

Where P, N and M are error, number of samples and model order respectively.

The fractional portion of FPE increases with M and accounts for the inaccuracies in estimating the parameters. The other criterion is called Akaike's information criterion (AIC). It is:

$$\text{AIC} = N \ln P^2 + 2P \qquad (3.3b)$$

The first criterion tends to have a minimum at values of M that are less than the model order and the second one tends to overestimate model order.

The above criteria were calculated for electrocardiogram signal and the results were plotted in Figure 2. As shown in Figure 2(a), the error decreases but there is no definitive slope change. The largest decrease occurs from order 1 to 2 and the error does not seem to decrease significantly with orders greater than 11. For FPE (Figure 2(b)) and AIC (Figure 2(c)) plots, the error does not decrease much with orders greater than 11. Thus, the order can be approximately 10. The Levinson-Durbin algorithm was used to calculate the AR parameters with order 10 for heart pulse. These parameters were used as features.

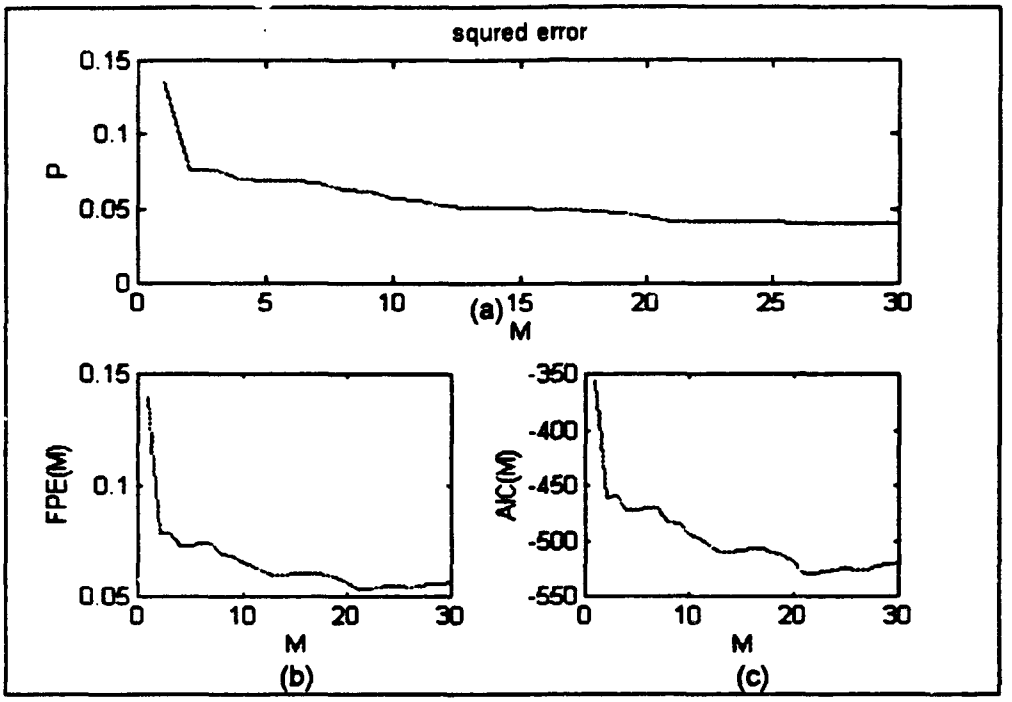


Figure 2. The different criteria for heart pulse versus model order (M): (a) error; (b) FPE; (c) AIC.

3.4 Cross-covariance and cross-correlation functions

In general, it may be necessary to study the interactions between two processes with possibly different scales of measurement or different variances. In polygraph where time series data are generated from more than one channel at a time, features like cross-correlation which contain information about relationships between the channels are extracted. The cross covariance(C_{xy}) and cross correlation function (r_{xy}) are defined as following:

$$C_{xy}(k) = \frac{\sum_{k=1}^{N-1}(X(n)-m_x)(Y(n+k)-m_y)}{N} \qquad [k = 0,1,....(N-1)] \quad (3.4a)$$

$$r_{xy} = Cxy / \sqrt{[Cxx(0)Cyy(0)]} \qquad (3.4b)$$

where $m_x = \sum_{k=1}^{N} \frac{X(n)}{N}$ $m_y = \sum_{k=1}^{N} \frac{Y(n)}{N}$ (3.4c)

$Cxx(0)$ and $Cyy(0)$ are the variances of observations on X and Y respectively.

This estimate is asymptotically unbiased. However, the variance of the estimate depends on the auto correlation functions of the two components. Therefore, for moderately large values of N it is possible for two series, which are actually uncorrelated, to give rise to large cross-correlation coefficients which are actually spurious. Thus, both series should first be filtered to convert them to white noise before computing the cross-correlation function [8].

In order to determine the relationship between the upper respiratory and heart rate, the cross correlation between them was calculated. Figure 3 shows the cross correlation between heart pulse and upper respiratory for a control and a relevant question for two different deceptive and non deceptive cases.

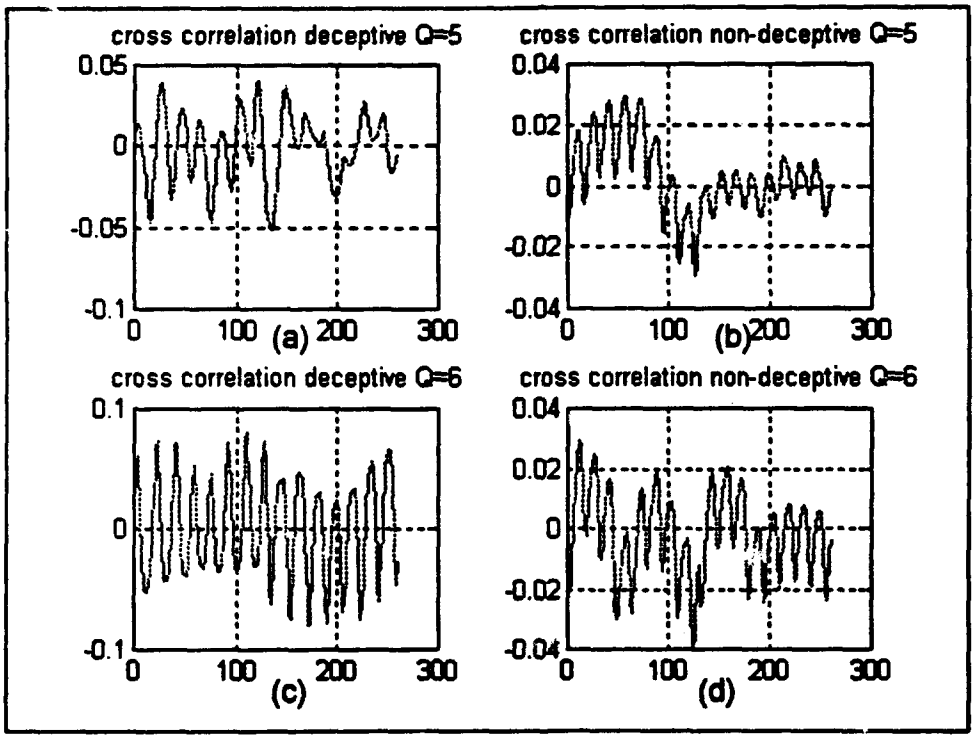


Figure 3. Cross correlation between upper respiratory and heart pulse before modeling. (a) and (b) 90 seconds after relevant question 5. (b) and (c) 90 seconds after control question 6.

3.5 Whitening filter

For a given process {x(n)}, the innovation process {v(n)} is defined as a white noise process such that {v(n)} can be determined from the signal {x(n)} by the whitening filter. The innovations representation of a random process is a powerful analytic tool. The innovation process makes the interpretation of the original process simpler than the original signal. Yet both processes contain the same statistical information. In other words, there is no loss of information as a result of the transformation.

As stated in section 3.4, it is possible for two series, which are actually uncorrelated, to give rise to large cross-correlation coefficients which are actually spurious. Thus, the series should first be filtered to convert them to white noise before computing the cross-correlation function. The AR parameters were used to design the whitening filter. Then, the heart pulse signal was filtered to convert it to white noise.

When the time series is white noise and purely random, the neighboring points of the ACF are uncorrelated. In order to compare the whitening filter output and the theoretical white noise, both the output of the whitening filter and its auto correlation for electrocardiogram were plotted in Figure 4. It is seen that the auto correlation shows high correlation for lag zero (k=175) and small correlation for other lags as it expected.

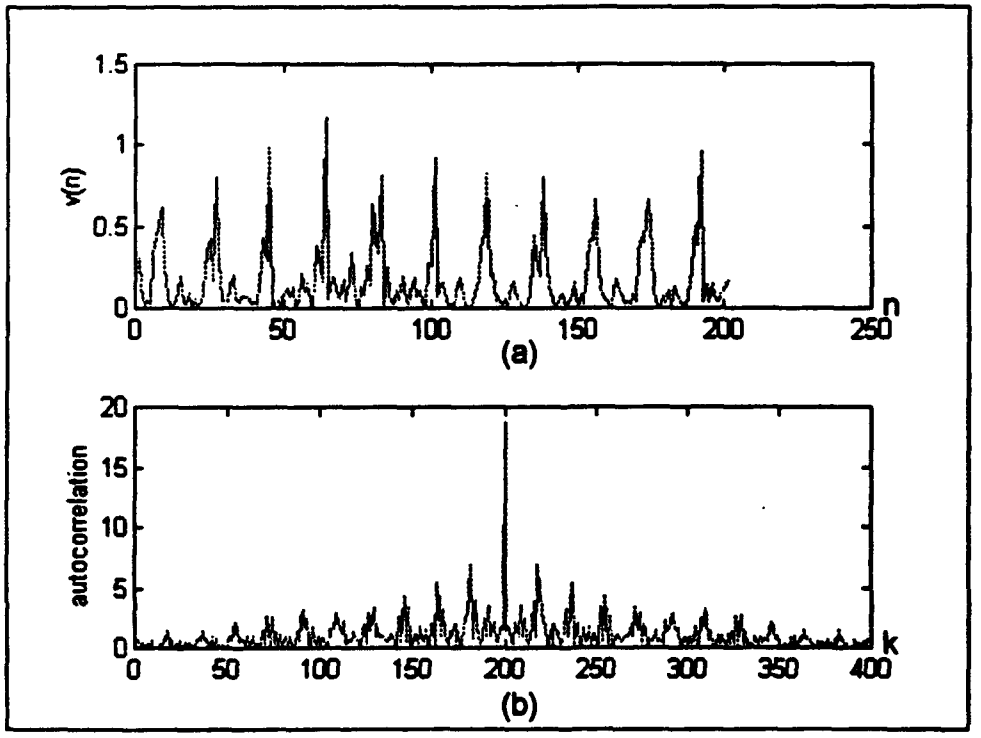


Figure 4. Plots of (a) white noise (output of the whitening filter); (b) auto correlation of the white noise.

The heart pulse and its innovation process (pre whitening filter output) contain the same information. The results of cross-correlation between upper respiratory and heart rate signals after pre whitening are shown in figure 5. It can be seen that the cross-correlation after modeling is similar to the cross correlation before modeling (Figure 2) with less spurious peaks. The maximum and minimum value of cross correlation and their lags were considered as potential features in correlation domain. As presented in figure 5 (b), heart pulse and upper respiratory channels are positively correlated after the 30 to 90 lags (1-3 seconds) and are negatively correlated after 130 lags (4.3 seconds).

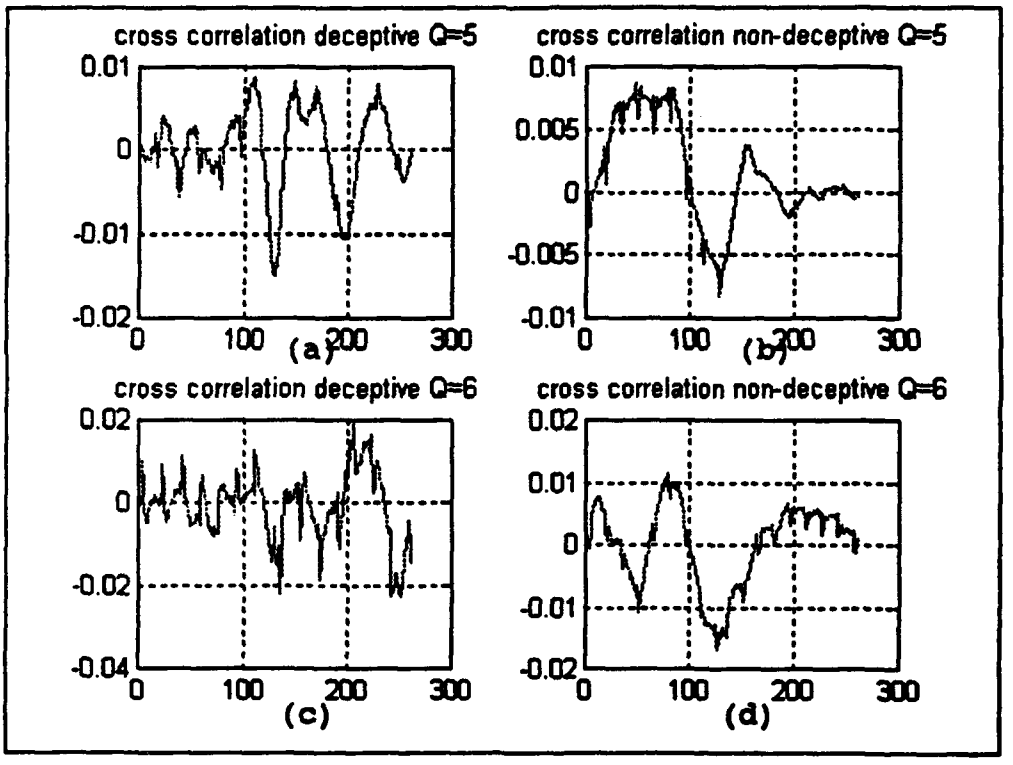


Figure 5. Cross correlation between heart pulse and upper respiratory after modeling for (a) and (b) 90 seconds after relevant question 5. (b) and (c) 90 seconds after control question 6.

3.6 Spectral Analysis

In this section the frequency properties of the polygraph signals such as power spectrum and cross spectral density are analyzed. The cross-correlation and cross spectral density are the tools for examining the relationships between two signals in the time and frequency domains respectively. The power spectrum shows how the variance of the signal is distributed with frequency. The total area underneath the spectrum curve is equal to the variance of the signal. A peak in the spectrum indicates an important contribution to the variance at different frequencies.

The estimated spectrum for different channels were plotted on linear scale in Figure 6 and on logarithmic scale in Figure 7. For spectrum showing large variations in power, a logarithmic scale makes it possible to show more detail over a wide range. However, this exaggerates the visual effects of variations where the spectrum is small. It is often easier to interpret the spectrum plotted on a linear scale than logarithmic scale.

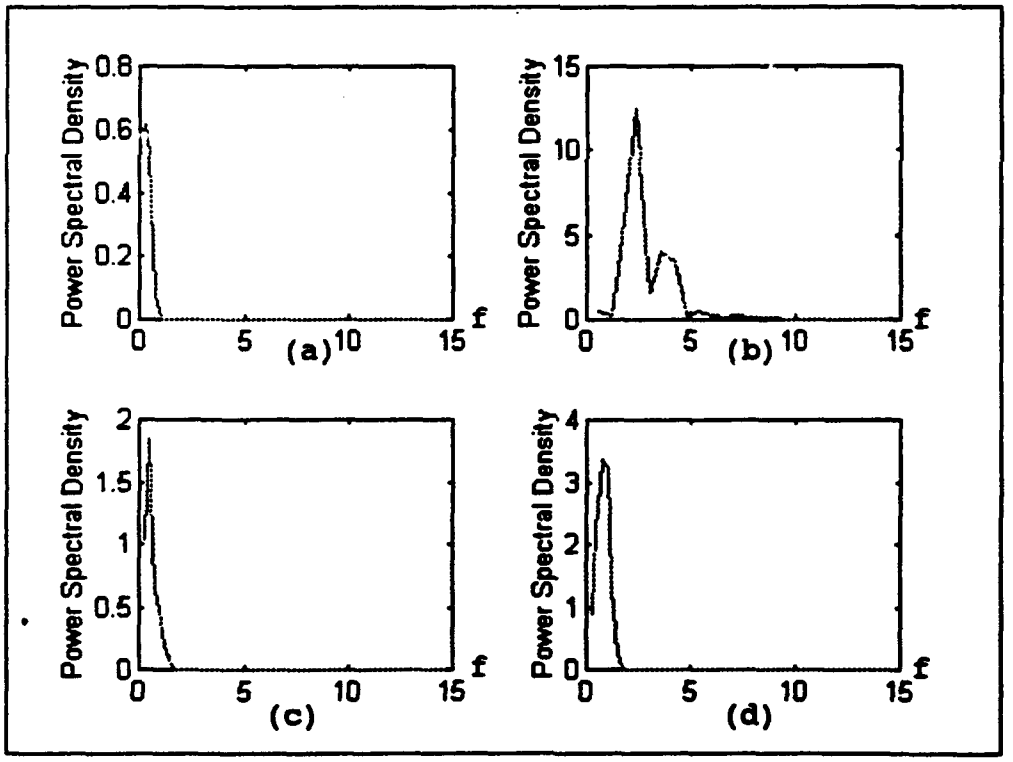


Figure 6. Frequency contents of four polygraph signals on linear scale. (a) GSR for 480 samples, (b) heart pulse for 200 samples, (c) and (d) lower and upper respiratory for 480 samples.

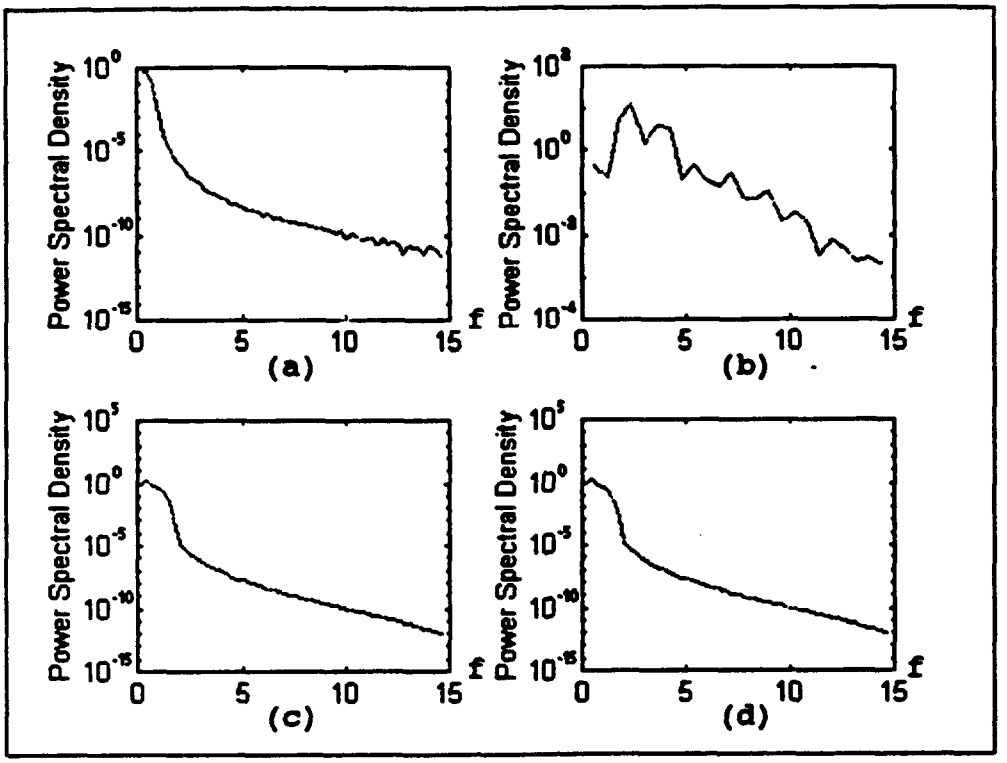


Figure 7. Frequency contents of four polygraph signals on logarithmic scale. (a) GSR for 480 samples, (b) heart pulse for 200 samples, (c) and (d) lower and upper respiratory for 480 samples.

Figure 7 shows for GSR the variance is concentrated at low frequencies indicating a trend or non-stationary behavior. The spectrum for heart pulse signal shows the presence of harmonics with a large peak at fundamental frequency of $f = 2$ Hz and related peaks at 2f, 3f, ...These multiples of the fundamental indicate the non sinusoidal character of the main cyclical component.

The correlation between two signals can be described in the frequency domain by their cross amplitude, phase spectra or the squared coherency. The coherency measures the linear correlation between the two components of the two channels at frequency f. The closer the coherency is to one, the more closely related are the two signals at frequency f.

The MATLAB function *spectrum.m* finds the cross-spectrum and coherency between upper respiratory and electrocardiogram and are shown in Figure 8. Their cross spectrum shows a large peak at $f = 2$ Hz. Maximum cross spectral density and the magnitude of cross spectral density and coherency at fundamental frequency and the second harmonic were considered as features in frequency domain.

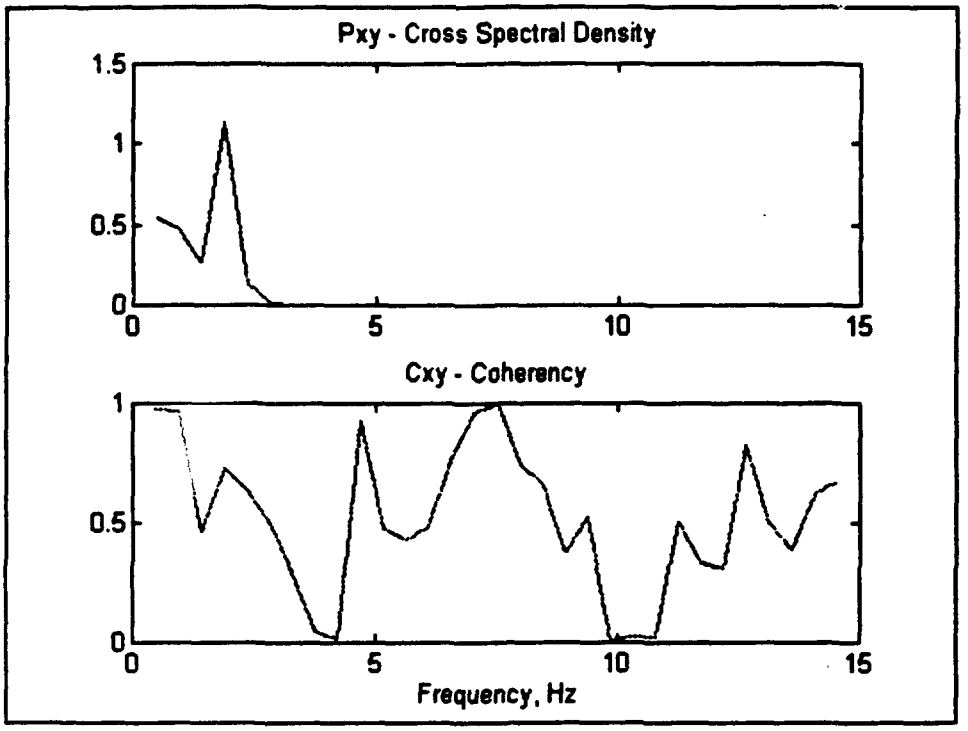


Figure 8. Plots of coherency and cross spectral density between heart pulse and upper respiratory signals.

3.7 Integrated spectral distance

This section describes how to obtain a feature in the frequency domain called integrated spectral difference. This feature was introduced by Martin and Pounds [12]. Other features are calculated separately for each control, relevant and irrelevant questions. The integrated spectral distance is calculated in a different way than the other features. This feature is calculated by taking the difference between the cumulative values of the power spectral density for each relevant and its closest control question. The integrated spectral distance measures the distance between a control and a relevant question directly. Figure 9 shows the cumulative spectral density for a control and a relevant question. The maximum, the frequency where this maximum happens and the area underneath were considered as features.

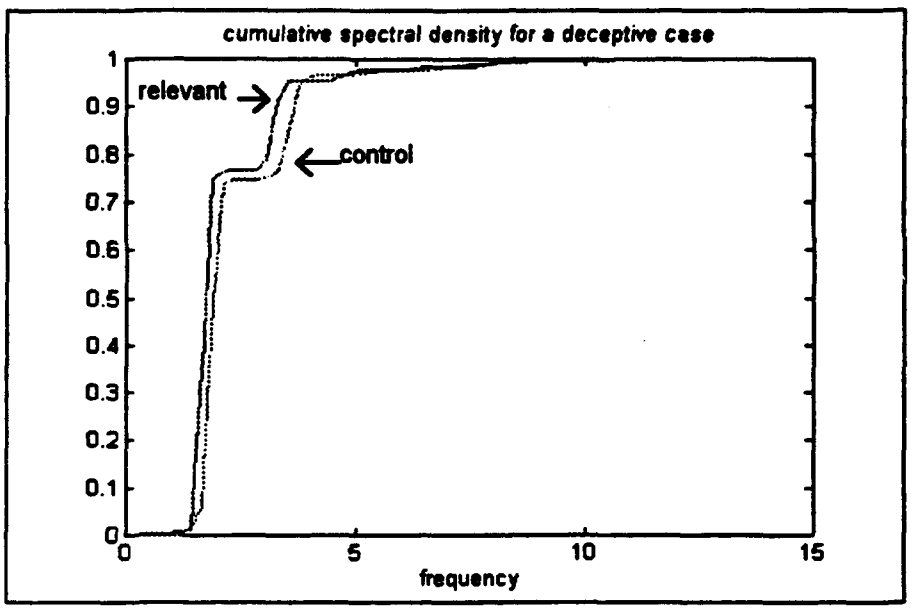


Figure 9. Cumulative integrated spectral density for a control question and relevant question of the heart pulse signal.

3.8 Frequency and Correlation Domain Features

Table 1 summarizes the frequency and correlation features explained in the above sections.

Feature	Channel
Maximum cross correlation	between 2 & 6
Lag of maximum cross correlation	between 2 & 6
Minimum cross correlation	between 2 & 6
Lag of minimum cross correlation	between 2 & 6
Spectral value at fundamental frequency	2
Spectral value at fundamental frequency	6
Spectral value at (fundamental frequency of channel 2) *2	2
Spectral value at (fundamental frequency of channel 6) *2	6
Maximum cross spectral density	between 2 & 6
Coherency at fundamental frequency of channel 2	between 2 & 6
Coherency (at fundamental frequency of channel 2)*2	between 2 & 6
Fundamental frequency	2
Fundamental frequency	5
Maximum or minimum integrated spectral difference	1
Frequency of the maximum integrated spectral difference	1
Area underneath integrated spectral difference	1
maximum or minimum integrated spectral difference	2
Frequency of the maximum integrated spectral difference	2
Area underneath integrated spectral difference	2
Autoregressive parameter	2

Table 1. Frequency and correlation domain features.

4 Feature extraction

4.1 Preprocessing

This chapter explains the steps taken in feature extraction algorithm. In polygraph tests, four physiological responses are measured. These responses are: upper respiratory, lower respiratory, galvanic skin response (GSR) and electrocardiogram. These four polygraph responses are processed into six channels. A low frequency electrocardiogram channel is produced by lowpass filtering the electrocardiogram channel. A high frequency electrocardiogram channel is produced by highpass filtering it. The high frequency electrocardiogram, called heart pulse, the low frequency electrocardiogram, called blood volume and derivative of the low frequency electrocardiogram are used instead of one electrocardiogram channel. To eliminate the noise and any trend, all the signals are filtered and detrended. For more information about the filtering and detrending refer to Jacobs [10].

4.2 Feature Selection

Many of the time domain features were selected based on the examiners' suggestions. However, many of the standard statistical features were also considered as potential features. For more information about time domain features refer to Jacobs [10]. The selected features and the channels which they were extracted from are listed below.

Features	Channel
1) Mean	1 ,2 , 3, 4, 5 , 6
2) Standard deviation	1 ,2 , 3, 4, 5 , 6
3) Minimum	1 ,2 , 3, 4, 5 , 6
4) Maximum	1 ,2 , 3, 4, 5 , 6
5) Curve length	1 ,2 , 3, 4, 5 , 6
6) Mean of derivative	1 ,2 , 3, 4, 5 , 6
7) Median of derivative	1 ,2 , 3, 4, 5 , 6
8) Average amplitude of peaks	2 , 5 , 6
9) Minimum amplitude of peaks	2 , 5 , 6
10) Derivative of amplitudes of peaks	2 , 5 , 6
11) Number of peaks	2 , 5 , 6
12) Minimum subtracted from maximum	1 ,2 , 3, 4, 5 , 6
13) Inhalation/exhalation	5 , 6
14) ratio of inhalation/exhalation before and after a question is asked	5 , 6
15) Fundamental frequency	2, 5
16) Maximum cross correlation	between 2 and 6
17) Lag of maximum cross correlation	between 2 and 6
18) Minimum cross correlation	between 2 and 6
19) Lag of minimum cross correlation	between 2 and 6
20) Spectral value at fundamental frequency	between 2 and 6
21) Spectral value at second harmonic	between 2 and 6
22) Maximum cross spectral density	between 2 and 6
23) Coherency at fundamental frequency	between 2 and 6
24) Coherency at second harmonic	between 2 and 6
25) Autoregressive parameters(AR)	2
26) Maximum or minimum integrated spectral difference (ISD)	1, 2
27) Frequency of maximum ISD	1, 2
28) Area under ISD	1, 2

4.3 Feature Extraction Algorithm

All features are extracted for 10 relevant, irrelevant and control questions except features 26, 27 and 28 that are extracted for each relevant and its closest control question. The program called fextract.m extracts all the basic features for each question on each chart for about 18 non-deceptive and 51 deceptive cases. Due to the small number of non-deceptive cases, each chart for a subject was used as a separate case. By doing this 50 non-deceptive and 150 deceptive files were created.

The test format used in this project is MGQT format. It is a type of control question test in which relevant, irrelevant and control questions are asked in a specific order. Each polygraph test is made of three and in very rare cases four charts for each case. The order in which the questions are asked is changed in the third and fourth charts and sometimes in the second chart. The feature extraction routine needs to have the control, relevant and irrelevant questions labeled. Therefore, for each polygraph chart a complementary chart called question file was created which contains a matrix called Q. The first row of this matrix contains the relevant, the second row the irrelevant and the third row the control questions respectively.

Fragments of each signal are selected before features are extracted. These fragments are shown in Table 2. Start and end points given in the table refer to the time elapsed after the question is asked. A vector of features for each file is created by the program feature.m which is called by fextract.m program. The program first executes all of the processing routines and then extracts the features for each question in the file. The features are extracted for the appropriate time segment (see Table 2) of six channels for each polygraph file. The time segment is created by taking a sample of time series starting several seconds after a question is asked and continuing for a number of seconds.

Channel description	Channel	Start	End
Galvanic Skin conductivity(GSR)	1	2 sec.	14 sec.
High frequency electrocardiogram	2	2 sec.	9 sec.
Low frequency electrocardiogram (LC)	3	2 sec.	18 sec.
Derivative of low frequency electrocardiogram (DLC)	4	0 sec.	8 sec.
Lower Respiratory (LR)	5	2 sec.	18 sec.
Upper Respiratory (UR)	6	2 sec.	18 sec.

Table 2. Time fragment used in feature extraction

The feature extraction algorithm provides a 960 dimensional vector for each file. The features were extracted for the 150 deceptive and 50 non deceptive files and saved in a 960 by 200 matrix called " M". In order to classify subjects using the difference between control and relevant responses, and to make the feature vector smaller, the features were combined according to the following method: for each feature *i* except features 26, 27,28 from each subject *j* compute:

1) The average control responses *AvCij*
2) The average relevant responses *AvRij*
3) The maximum and minimum control responses *MaxCij* and *MinCij*
4) The maximum and minimum relevant responses *MaxRij* and *MinRij*

The feature vector components for feature i are then:

$$1) Fij(1) = AvRij - AvCij$$

$$2) Fij(2) = \frac{AvRij - AvCij}{AvRij + AvCij}$$

$$3) Fij(3) = MaxRij - MaxCij$$

$$4) Fij(4) = MinRij - MinCij$$

$$5) Fij(5) = MaxRij - MinCij$$

$$6) Fij(6) = MinRij - MaxCij$$

$$7) Fij(7) = \frac{MaxRij}{MaxCij}$$

For features 26, 27, 28 from each subject *j* compute:

1) The average of relevant-control responses *Av(RC(ij)*
2) The maximum of relevant-control responses *Max(RC(ij)*
3) The minimum of relevant-control responses *Min(RC(ij)*

The feature vector components for feature *i* are then:

$$1) F_{ij}(1) = Av(RC(_{ij}))$$

$$2) F_{ij}(2) = Max(RC(_{ij}))$$

$$3) F_{ij}(3) = Min(RC(_{ij}))$$

The above procedure is executed by program called *procesf.m* which creates a 669 by 200 dimensional matrix called "F". In order to run the classifier program, the matrix F was divided into three 100 (50 deceptive and 50 non-deceptive) sets of matrices called set1, set2 and set3. These sets are made of 50 non-deceptive cases common in all three sets and three 50 different deceptive sets, called deceptive 1, deceptive 2 and deceptive 3 respectively. The list of the files used in the set1, set2 and set3 are shown in Table 3 in Appendix A.

5 Results

5.1 Frequency Domain Clustering

Classifier is the final stage in a pattern recognition system. The classifier assigns each input to one of the classes. The classifier could be designed after studying the distribution of samples in each class. The KNN classifier was used in this study because of the following:

1) The uncertainty about the shape of deceptive and non deceptive clusters and their sample distributions.
2) The possibility that the samples for one class cluster around more than one point in space.

The frequency domain features did not create a separate distribution of samples for deceptive and non deceptive classes. However, the combination of frequency and time domain features resulted in more distinct clusters. Figure 10 and 11 show the examples of sample distribution (clustering) for non deceptive (x) and deceptive (+) classes.

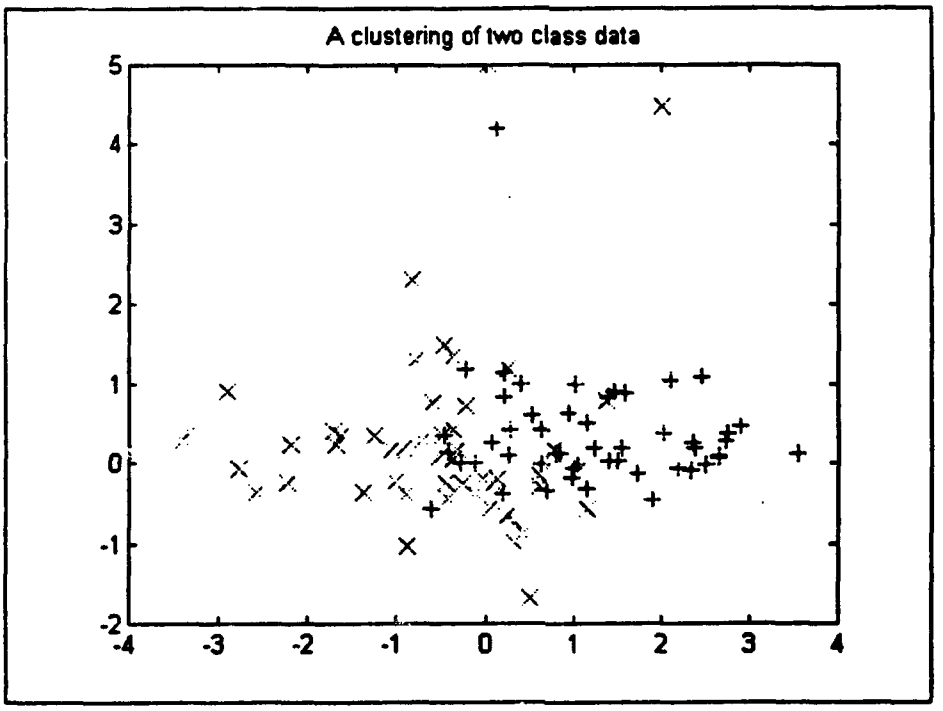


Figure 10. Plot of maximum of GSR versus maximum of Upper Respiratory.

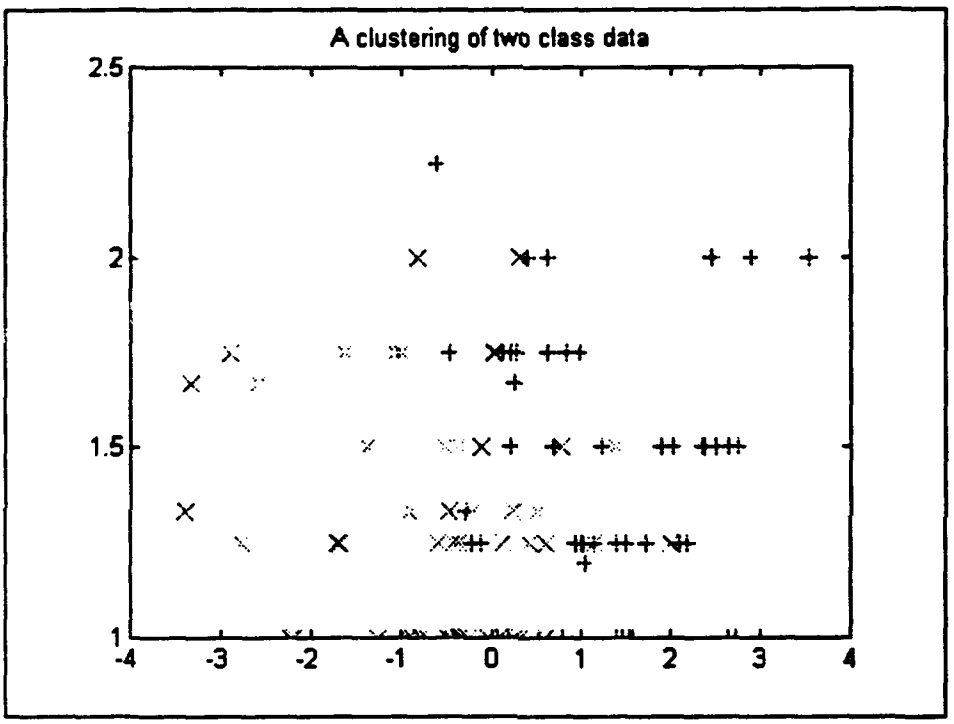


Figure 11. Plot of maximum of GSR versus frequency of maximum integrated spectral difference of GSR.

5.2 Discussion

The 669 features are more than can be used by any classification techniques. Thus, the classification program and the scatter measurement program were run for each feature in each set individually. The results of the first experiment were examined and compared to determine the features which were the best discriminators between deceptive and non-deceptive subjects. After comparing the results, the 30 features with the highest accuracy rate and common in all three sets were selected. These best features were listed in Table 3.

The second experiment used the combination of two features out of the best 30 features. The results for the best 30 features were examined for each set separately. The set3 always had a better performance than the other two sets. However, in order to be consistent, the best features common in all three sets were selected as the 30 best features. More features were added for combination of three and four. The results are shown in Table 4 and 5 in Appendix A.

As it was discussed before, the classifier was used to compare the effectiveness of the single features and to choose the combination of the best features. Changing the classifier parameters such as K might change the results of the classification. However, it is not practical to change all parameters at the same time. Therefore, the classifier was used with the fixed parameters of K=5 and m=2. After selecting the final feature set, theses parameters were changed to find the best classification.

No	feature	Description	Channel	Method
1	10mean	mean	GSR	1
2	10curve	curve length	GSR	2
3	10med_dif	median of the derivative	GSR	1
4	10max_min	minimum subtracted from the maximum	GSR	2
5	10max	maximum of the signal	GSR	1
6	10mdif	mean of derivative	GSR	3
7	20curve	curve length	Heart pulse	1
8	20ampcard	amplitude of the peaks	Heart pulse	1
9	20max_min	minimum subtracted from the maximum	Heart pulse	4
10	20max	maximum of the signal	Heart pulse	4
11	20min	minimum of the signal	Heart pulse	1
12	30med_dif	median of the derivative	Blood pressure	3
13	30max	maximum of the signal	Blood pressure	1
14	40mean	mean	Derivative of Blood pressure	1
15	40max	maximum of the signal	Derivative of Blood pressure	1
16	50curve	curve length	Lower Respiratory	6
17	50ampr	amplitude of the peaks	Lower Respiratory	2
18	50peaknumr	number of the peaks	Lower Respiratory	5
19	50ie	inhalation divided by exhalation	Lower Respiratory	5
20	50max_min	minimum subtracted from the maximum	Lower Respiratory	2
21	50max	maximum of the signal	Lower Respiratory	6
22	60max_min	minimum subtracted from the maximum	Upper Respiratory	2
23	60max	maximum	Upper Respiratory	3
24	10std	standard deviation	GSR	2
25	20std	standard deviation	Heart pulse	1
26	50std	standard deviation	Upper Respiratory	6
27	20armod1	auto regressive parameter	Heart pulse	7
28	26psdcoh1	max cross spectral density	Heart pulse, Lower Respiratory	1
29	10isd1	frequency of maximum integrated spectral difference of control-relevant pair	GSR	1*
30	20isd1	area under integrated spectral difference	Heart pulse	3*

Methods: 1=Difference of Averages, 2=Normalized Average, 3=Max-Max, 4=Min-Min,

5=Max-Min, 6=Min-Max, 7=Max/Min , 1*=Average of relevant-control pairs, 3*=Max of relevant-control pair.

Table 3. 30 best selected Features

Conclusion

The classification results improved consistently by increasing the number of features. The best features are {5 9 21 23} and {5 21 23 29}with 81 and 80 percent correct classification respectively. These features are maximum of GSR(5), difference between maximum and minimum of heart pulse(9), maximum of lower respiratory(21), maximum of upper respiratory(23) and frequency of maximum integrated spectral difference of control-relevant pair for GSR(29).

The best features are simple and obvious features such as maximum and minimum of the polygraph signals. In other words, the features that an examiner can see are the best discriminators between deceptive and non deceptive.

It is important to notice that the best features are the combination of features from all 4 different GSR, heart pulse, lower and upper respiratory. As expected, each subject shows reaction to different channels. Therefore, the combination of all channels is the best representative of deception.

Another point to notice is that the set3 has better classification results than the other two sets. For example, the features {9 14 19 24} and {5 21 23 29}show 87.4 and 86.6 percent correct classification for set3. The data in set3 is made of 50 non deceptive common in all three sets and 50 deceptive cases. This set of deceptive cases, called deceptive 3, are the Acxiton files listed in Table 3 in Appendix A. It is possible that there is some characteristic in these deceptive files that results in better classification.

As stated before, due to the small number of non-deceptive cases available, each chart for a subject was used as a separate case. After classifying the charts, the charts for each case were combined in a way that each case was assigned to the class that the majority of the charts belong to. Using this method, the classification results improved from 81 percent to 85.6 percent for set1 and set2 and from 87 percent to 91 percent for set3. The final result is included in appendix A.

References

[1] Dale E. Olsen, et. al., "Recent developments in polygraph testing: A research review and evaluation - A technical memorandum, " Washington, DC: US Government Printing Office 1983.

[2] John E. Reid and Fred E. Inbau, Truth and Deception: The Polygraph (" Lie Detector ") Technique, The Williams & Wilkins Company, Baltimore, Md., 1966

[3] Michael H. Capps and Norman Ansley, "Numerical Scoring of Polygraph Charts: What Examiners Really Do", Polygraph, 1992, 21, 264-320

[4] Personal communication with Richard Petty (polygraph examiner), June 1993

[5] J.M.Keller, M.R. Gray and J.A. Givens, "A fuzzy K nearest neighbor algorithm", IEEE Trans. on syst.Man Cybernetics, vol SMC-15, No.14

[6] J.C. Bezdek and Siew K.Chuan, "Generalized K nearest neighbor rules, Fuzzy sets and System vol 18(1986).

[7] Rabiner and Schafer, "Digital Processing of Speech Signals", p141.

[8] Jenkins and Watts, 1968, p. 340.

[9] Richard Shiavi, "Introduction to Applied Statistical Signal Analysis", P357.

[10] Eric Jacobs ,"Time Domain Features of Polygraph Data", Masters Project Report, San Jose State University, Fall 1993.

[11] Shahab Layeghi, "Pattern Recognition of the Polygraph Using Fuzzy Set Theory", Masters Project report, San Jose State University, Fall 1993.

[12] R. Douglas Martin, Ph.D. and Christopher B. Pounds, Polygraph Reliability, the Department of Statistics University of Washington Seattle, Washington 98195 October 1,1991- September 30, 1992.

Appendices

Appendix A

Tables

FILE NAME		FUNDAMENTAL FREQUENCY (Hz) CHANNEL : Heart pulse, WINDOW: 120 S			
QQAV53P6.021	relevant =	1.3636	1.3636	1.3636	1.4286
	control =	1.2500	1.5000		
QQAV53P6.031	relevant =	1.5000	1.3636	1.3043	1.3636
	control =	1.4286	1.3636	1.3636	1.4286
QQBQ4SHI.011	relevant =	2	2	2	2
	control =	2	2		
QQBQ4SHI.021	relevant =	1.7647	1.7647	1.7647	1.8750
	control =	1.8750	1.76		
QQBQ4SHI.031	relevant =	1.7647	1.7647	1.7647	1.7647
	control =	0.8571	1.7647	1.7647	1.6667
QQBSS7WT.011	relevant =	1.5000	1.5000	1.5000	1.3636
	control =	1.5789	1.4286		
QQBSS7WT.021	relevant =	1.5000	1.4286	1.4286	1.4286
	control =	1.5000	1.4286		
QQBSS7WT.031	relevant =	1.4286	1.5000	1.4286	1.3636
	control =	1.4286	1.5000	1.4286	1.5000

Table 1. Fundamental frequency for non-deceptive files for 120 seconds for heart pulse.

FILE NAME	FUNDAMENTAL FREQUENCY(Hz) CHANNEL : CARDIO, WINDOW: 120 S			
QQ9SOW8L.021	relevant =	1.7647	1.6667	1.5789	1.6667
	control =	1.5789	1.5789		
QQ9SOW8L.031	relevant=	1.5789	1.5789	1.6667	1.6667
	control =	1.8750	1.6667	1.7647	1.5789
QQ9SQIK9.011	relevant =	1.5789	1.5000	1.5000	1.5789
	control =	1.5789	1.5000		
QQ9SQIK9.021	relevant =	1.3043	1.5789	1.5789	1.4286
	control =	1.5789	1.5789		
QQ9SQIK9.031	relevant =	1.5000	1.5000	1.6667	
	control =	1.4286	1.2000	1.5789	1.5789
QQ9W0B9F.011	relevant =	1.5000	1.4286	1.5000	1.5000
	control =	1.4286	1.5789		
QQ9W0B9F.031	relevant=	1.4286	1.5000	1.4286	1.4286
	control =	1.5000	1.4286		
QQ9W0B9F.041	relevant =	1.4286	1.3636	1.4286	1.5000
	control =	1.4286	1.3636		
QQ9U4FMU.011	relevant =	1.5789	1.6667	1.6667	1.6667
	control =	1.6667	1.5789		

Table 2. Fundamental frequency for deceptive files for 120 seconds for heart pulse.

Non deceptive	Deceptive 1	Deceptive 2	Deceptive 3
QQ8R9OIO.011	QQ4Q1O83.011	QQ7LX5Q0.021	QQ8RAJ0C.011
QQ8R9OIO.021	QQ4Q1O83.021	QQ7LX5Q0.031	QQ8RAJ0C.021
QQ8R9OIO.031	QQ4Q1O83.031	QQ7MN2Y0.011	QQ8RAJ0C.031
QQ95LU1T.011	QQ4Q3MDC.011	QQ7MN2Y0.021	QQ9EUKVT.011
QQ95LU1T.021	QQ4Q3MDC.021	QQ7MN2Y0.031	QQ9EUKVT.021
QQ95LU1T.031	QQ4Q3MDC.031	QQ7TC5UF.011	QQ9EUKVT.031
QQAURNUS.021	QQ51DE36.011	QQ7TC5UF.021	QQ9IOOXO.021
QQAURNUS.031	QQ51DE36.021	QQ7TC5UF.031	QQ9IOOXO.041
QQAV53P6.011	QQ51DE36.041	QQ7TQVER.011	QQ9SOW8L.011
QQAV53P6.021	QQ6RQGH6.011	QQ7TQVER.021	QQ9SOW8L.021
QQAV53P6.031	QQ6RQGH6.021	QQ7TQVER.031	QQ9SOW8L.031
QQBQ4SHI.011	QQ6RQGH6.031	QQ7TVADC.011	QQ9SQIK9.011
QQBQ4SHI.021	QQ6RQGH6.041	QQ7TVADC.021	QQ9SQIK9.021
QQBQ4SHI.031	QQ6T711O.011	QQ7TVADC.031	QQ9SQIK9.031
QQBSS7WT.011	QQ6T711O.021	QQ7U2T4R.011	QQ9W0B9F.011
QQBSS7WT.021	QQ6T711O.031	QQ7U2T4R.021	QQ9W0B9F.031
QQBSS7WT.031	QQ6Z59IG.011	QQ7U2T4R.031	QQ9W0B9F.041
QQ7OXM60.021	QQ6Z59IG.021	QQ7YP7QU.011	QQ9U4FMU.011
QQ7RH0RO.011	QQ6Z59IG.031	QQ7YP7QU.021	QQ9U4FMU.021
QQ7RH0RO.021	QQ7PP9B9.011	QQ7YP7QU.031	QQ9U4FMU.031
QQ7RH0RO.031	QQ7PP9B9.021	QQ7YZOJ3.011	QQ9Y_SVF.011
QQ7R51P9.011	QQ7PP9B9.031	QQ7YZOJ3.021	QQ9Y_SVF.021
QQ7R51P9.021	QQ7PDU1X.011	QQ7YZOJ3.031	QQ9Y_SVF.031
QQ7R51P9.031	QQ7PDU1X.021	QQ8_0DPT.011	QQ9YH3QF.011
QQ9TDSP3.011	QQ7PDU1X.031	QQ8_0DPT.021	QQ9YH3QF.021
QQ9TDSP3.021	QQ7_PIPF.011	QQ8_0DPT.031	QQ9YH3QF.031
QQ9TDSP3.031	QQ7_PIPF.021	QQ8_0DPT.041	QQA2TT4C.011
QQA8OWOI.011	QQ7_PIPF.031	QQ8_2UQ9.011	QQA2TT4C.021
QQA8OWOI.021	QQ7_JT70.011	QQ8_2UQ9.021	QQA2TT4C.031
QQA8OWOI.031	QQ7_JT70.021	QQ8_2UQ9.031	QQA3HIRX.011
QQBT22O6.011	QQ7_JT70.031	QQ800IG6.011	QQA3HIRX.021
QQBT22O6.021	QQ738DYX.011	QQ800IG6.021	QQA3HIRX.031
QQBT22O6.031	QQ738DYX.021	QQ800IG6.031	QQA32UTF.011
QQBO9O_9.011	QQ738DYX.031	QQ82OIU9.011	QQA32UTF.021
QQBO9O_9.021	QQ75ULP9.011	QQ82OIU9.021	QQA32UTF.031
QQBO9O_9.031	QQ75ULP9.021	QQ82OIU9.031	QQA6U_IF.011
QQBC7PP6.011	QQ75ULP9.031	QQ82SUTX.011	QQA6U_IF.031
QQBC7PP6.021	QQ79_EYF.011	QQ82SUTX.021	QQA6U_IF.041
QQBC7PP6.031	QQ79_EYF.021	QQ82SUTX.031	QQAM4E3L.011
QQCHCK_O.011	QQ79_EYF.031	QQ860ZNU.011	QQAM4E3L.021
QQCHCK_O.021	QQ7BGDML.011	QQ860ZNU.021	QQAM4E3L.031
QQCHCK_O.031	QQ7BGDML.021	QQ860ZNU.031	QQARF2_X.011
QQCDTKP0.011	QQ7BGDML.031	QQ89U_ZR.011	QQARF2_X.021
QQCDTKP0.031	QQ7ETC8I.011	QQ89U_ZR.021	QQARF2_X.031
QQCDTKP0.041	QQ7ETC8I.021	QQ89U_ZR.031	QQAWA38X.011
QQCM5Y56.011	QQ7ETC8I.031	QQ8ATU26.011	QQAWA38X.021
QQCQQT8Y.011	QQ7JAQCS.011	QQ8ATU26.021	QQAWA38X.031
QQCQQT8Y.021	QQ7JAQCS.021	QQ8ATU26.031	QQAYXZGU.011
QQCQQT8Y.031	QQ7JAQCS.031	QQ8FGMVI.011	QQAYXZGU.021
QQCQQT8Y.041	QQ7LX5Q0.011	QQ8FGMVI.021	QQAYXZGU.031

Table 3. List of files used in this experiment. 50 non-deceptive cases and 50 deceptive cases from set1, set2 and set3 are listed in column 1 through 4 respective

Set	Features			accuracy
Set1	10	21	26	79.4
	5	11	23	77.6
	5	21	23	77.4
Set2	12	20	24	79.8
	19	24	30	78.6
	5	21	23	77.4
Set3	9	19	24	85.2
	5	23	29	82.4
	5	21	23	81.2
Average	5	23	29	78.2
	5	7	23	77.6
	5	21	23	77.3

Table 4. The three best features of combination of 3 for each set and their average.

Set	Features				accuracy
Set1	5	9	21	23	81.0
	5	11	21	23	80.2
	5	21	23	29	74.4
Set2	5	14	23	29	81.0
	5	9	21	23	79.4
	5	21	23	29	79.0
Set3	9	14	19	24	87.4
	5	21	23	29	86.6
	5	21	23	9	82.5
Average	5	9	21	23	81.0
	5	21	23	29	80.0
	5	21	23	11	79.8

Table 5. The three best features of combination 4 for each set and their average.

File	Membership	Defuzzified	Result
1.0000	0.2736	0	
2.0000	0.3339	0	
3.0000	0.5397	0	0
4.0000	0.5450	0	
5.0000	0.7423	1.0000	
6.0000	0.1732	0	0
7.0000	0.8901	1.0000	
8.0000	1.0000	1.0000	1 Misclassified
9.0000	0.5376	0	
10.0000	0.1742	0	
11.0000	0.4366	0	0
12.0000	0.3458	0	
13.0000	0.5145	0	
14.0000	0.5178	0	0
15.0000	0.1016	0	
16.0000	0	0	
17.0000	0	0	0
18.0000	0.1334	0	0
19.0000	0	0	
20.0000	0	0	
21.0000	0.2923	0	0
22.0000	0	0	
23.0000	0	0	
24.0000	0.1607	0	0
25.0000	0	0	
26.0000	0.4421	0	
27.0000	1.0000	1.0000	0
28.0000	0.3307	0	
29.0000	0.0583	0	
30.0000	0.4965	0	0
31.0000	0.3505	0	
32.0000	0.1181	0	
33.0000	0.2101	0	0

Table 6. Classification of the files in Set1.

File	Membership	Defuzzified	Result
34.0000	0.5970	0	
35.0000	0	0	
36.0000	0.1193	0	0
37.0000	0.3174	0	
38.0000	0.8117	1.0000	
39.0000	0.0997	0	0
40.0000	0.1889	0	
41.0000	0.4215	0	
42.0000	0.1635	0	0
43.0000	0.6474	1.0000	
44.0000	0	0	
45.0000	0.5495	0	0
46.0000	0.1115	0	0
47.0000	0	0	
48.0000	0.3986	0	
49.0000	0	0	
50.0000	0	0	0
51.0000	0.6709	1.0000	
52.0000	1.0000	1.0000	
53.0000	0.5297	0	1
54.0000	0.7245	1.0000	
55.0000	0.9200	1.0000	
56.0000	1.0000	1.0000	1
57.0000	0.9105	1.0000	
58.0000	0.9398	1.0000	
59.0000	0.5657	0	1
60.0000	0.8968	1.0000	
61.0000	1.0000	1.0000	
62.0000	0.2793	0	
63.0000	0.1088	0	0 Misclassified
64.0000	0.6245	1.0000	
65.0000	0.8643	1.0000	
66.0000	0.5054	0	1

Table 6. Continued.

File	Membership	Defuzzified	Result
67.0000	0.8498	1.0000	
68.0000	0.6969	1.0000	
69.0000	0.8397	1.0000	1
70.0000	0.2901	0	
71.0000	0.8291	1.0000	
72.0000	0.3982	0	0 Misclassified
73.0000	1.0000	1.0000	
74.0000	0.2463	0	
75.0000	0.8043	1.0000	1
76.0000	0.6676	1.0000	
77.0000	1.0000	1.0000	
78.0000	1.0000	1.0000	1
79.0000	1.0000	1.0000	
80.0000	0.7538	1.0000	
81.0000	1.0000	1.0000	1
82.0000	1.0000	1.0000	
83.0000	0.8378	1.0000	
84.0000	1.0000	1.0000	1
85.0000	0.8926	1.0000	
86.0000	0.5448	0	
87.0000	0.5751	0	0 Misclassified
88.0000	0.8273	1.0000	
89.0000	0.2945	0	
90.0000	0.9110	1.0000	1
91.0000	1.0000	1.0000	
92.0000	1.0000	1.0000	
93.0000	0	0	1
94.0000	0.2887	0	
95.0000	0.2079	0	
96.0000	0.5793	0	0 Misclassified
97.0000	1.0000	1.0000	
98.0000	0.7971	1.0000	
99.0000	0.8708	1.0000	1
100.0000	1.0000	1.0000	1

Table 6. Continued.

File	Membership	Defuzzified	Result
1.0000	0.2579	0	
2.0000	0.1307	0	
3.0000	0	0	0
4.0000	0.2652	0	
5.0000	0.4345	0	
6.0000	0.1175	0	0
7.0000	1.0000	1.0000	
8.0000	0.7086	1.0000	1 Misclassified
9.0000	0.2856	0	
10.0000	0.2745	0	
11.0000	0.3056	0	0
12.0000	0.2720	0	
13.0000	0.5019	0	
14.0000	0.8871	1.0000	0
15.0000	0.0912	0	
16.0000	0	0	
17.0000	0	0	0
18.0000	0.8334	1.0000	1 Misclassified
19.0000	0	0	
20.0000	0	0	
21.0000	0.5483	0	0
22.0000	0	0	
23.0000	0	0	
24.0000	0.1535	0	0
25.0000	0.4955	0	
26.0000	0.1013	0	
27.0000	1.0000	1.0000	0
28.0000	0.3788	0	
29.0000	0.1638	0	
30.0000	0.0905	0	0
31.0000	0	0	
32.0000	0.1431	0	
33.0000	0.0937	0	0

Table 7. Classification of the files in set2.

File	Membership	Defuzzified	Result
34.0000	0	0	
35.0000	0	0	
36.0000	0.1281	0	0
37.0000	0.3690	0	
38.0000	0.5734	0	
39.0000	0.1569	0	0
40.0000	0.3659	0	
41.0000	0.4124	0	
42.0000	0.1704	0	0
43.0000	0.4251	0	
44.0000	0.0664	0	
45.0000	0.5356	0	0
46.0000	0.5084	0	0
47.0000	0.1735	0	
48.0000	0.7512	1.0000	
49.0000	0.5115	0	
50.0000	0.0976	0	0
51.0000	0.6361	1.0000	
52.0000	0.8482	1.0000	1
53.0000	0.3471	0	
54.0000	0.8822	1.0000	
55.0000	1 0000	1.0000	1
56.0000	1.0000	1.0000	
57.0000	1.0000	1.0000	
58.0000	0.8730	1.0000	1
59.0000	0	0	
60.0000	0.0389	0	
61.0000	0.3643	0	0 Misclassified
62.0000	1.0000	1.0000	
63.0000	0.8174	1.0000	
64.0000	0.8875	1.0000	1
65.0000	0.7995	1.0000	
66.0000	0.5919	0	
67.0000	0.7533	1.0000	1

Table 7. Continued.

File	Membership	Defuzzified	Result
68.0000	0.7337	1.0000	
69.0000	0.8524	1.0000	
70.0000	0.8602	1.0000	1
71.0000	0.2217	0	
72.0000	1.0000	1.0000	
73.0000	0.1268	0	0 Misclassified
74.0000	0.8860	1.0000	
75.0000	0.2121	0	
76.0000	0.1684	0	
77.0000	0.6903	1.0000	0 Misclassified
78.0000	0.7680	1.0000	
79.0000	0.8735	1.0000	
80.0000	0.8013	1.0000	1
81.0000	0.1748	0	
82.0000	0.5428	0	
83.0000	0.8496	1.0000	0 Misclassified
84.0000	0.3444	0	
85.0000	0.8298	1.0000	
86.0000	0.8590	1.0000	1
87.0000	0.6879	1.0000	
88.0000	0.9082	1.0000	
89.0000	0.6653	1.0000	1
90.0000	0.1636	0	
91.0000	0.8754	1.0000	
92.0000	0.8594	1.0000	1
93.0000	0.5185	0	
94.0000	0.4932	0	
95.0000	0.7802	1.0000	0 Misclassified
96.0000	0.8684	1.0000	
97.0000	0.8788	1.0000	
98.0000	1.0000	1.0000	1
99.0000	1.0000	1.0000	
100.0000	0.8669	1.0000	1

Table 7. Continued.

File	Membership	Defuzzified	Result
1.0000	0.3986	0	
2.0000	0.2845	0	
3.0000	0.2562	0	0
4.0000	0.2786	0	
5.0000	0.3226	0	
6.0000	0	0	0
7.0000	1.0000	1.0000	
8.0000	0.5055	0	
9.0000	0.1434	0	0
10.0000	0	0	
11.0000	0	0	0
12.0000	0.0691	0	
13.0000	0.4744	0	
14.0000	0.4708	0	0
15.0000	0	0	
16.0000	0	0	
17.0000	0	0	0
18.0000	0.4623	0	0
19.0000	0	0	
20.0000	0	0	
21.0000	0.2096	0	0
22.0000	0	0	
23.0000	0	0	
24.0000	0.0516	0	0
25.0000	0.2885	0	
26.0000	0.0981	0	
27.0000	0.9336	1.0000	0
28.0000	0.2254	0	
29.0000	0.1465	0	
30.0000	0.0680	0	0
31.0000	0	0	
32.0000	0	0	
33.0000	0.0939	0	0

Table 8. Classification of the files in Set3.

File	Membership	Defuzzified	Result
34.0000	0.3917	0	
35.0000	0	0	
36.0000	0	0	0
37.0000	0.1689	0	
38.0000	0.5220	0	
39.0000	0	0	0
40.0000	0.0969	0	
41.0000	0	0	
42.0000	0	0	0
43.0000	0.4810	0	
44.0000	0.3154	0	
45.0000	0.4552	0	0
46.0000	0.3285	0	0
47.0000	0.3690	0	
48.0000	0.5593	0	
49.0000	0.3522	0	
50.0000	0.2325	0	0
51.0000	1.0000	1.0000	
52.0000	0.9052	1.0000	
53.0000	0.8115	1.0000	1
54.0000	0.8397	1.0000	
55.0000	0.8754	1.0000	
56.0000	0.0930	0	1
57.0000	0.8330	1.0000	
58.0000	1.0000	1.0000	1
59.0000	1.0000	1.0000	
60.0000	1.0000	1.0000	
61.0000	1.0000	1.0000	1
62.0000	1.0000	1.0000	
63.0000	0.6496	1.0000	
64.0000	0.5075	0	1
65.0000	0.0823	0	
66.0000	0.7810	1.0000	
67.0000	0.2356	0	0 Misclassified

Table 8. Continued.

File	Membership	Defuzzified	Result
68.0000	1.0000	1.0000	
69.0000	1.0000	1.0000	
70.0000	1.0000	1.0000	1
71.0000	1.0000	1.0000	
72.0000	1.0000	1.0000	
73.0000	1.0000	1.0000	1
74.0000	1.0000	1.0000	
75.0000	1.0000	1.0000	
76.0000	1.0000	1.0000	1
77.0000	1.0000	1.0000	
78.0000	1.0000	1.0000	
79.0000	1.0000	1.0000	1
80.0000	0.6068	1.0000	
81.0000	0.9054	1.0000	
82.0000	0.4134	0	1
83.0000	1.0000	1.0000	
84.0000	0	0	
85.0000	0.2914	0	0 Misclassified
86.0000	1.0000	1.0000	
87.0000	1.0000	1.0000	
88.0000	0.8786	1.0000	1
89.0000	0.9018	1.0000	
90.0000	1.0000	1.0000	
91.0000	1.0000	1.0000	1
92.0000	1.0000	1.0000	
93.0000	0.9135	1.0000	
94.0000	0.8292	1.0000	1
95.0000	0.7423	1.0000	
96.0000	1.0000	1.0000	
97.0000	0.0902	0	1
98.0000	0.2564	0	
99.0000	0	0	
100.0000	0.4387	0	0 Misclassified

Table 8. Continued.

Appendix B

Programs

```
function v=armod(var,M)

% This function finds the autoregressive parameter fo the signal
% and then prewhitens the signal using the prewhiten filter.
% Recursive Levinston and durbin algorithm is used to find the AR parameters

% To use the function the user should enter the signal and the AR model order
% eg armod(variable, model order)

Fs=30;                                  %sampling frequency


r=xcorr(var,'biased');                  %rx(0) is at index K
K=length(var);
rx=r(K:K+M+1);                          %rx(0),rx(1),..rx(M)


% Estimate the reflection coefficients

a(1,1)=1;
P=rx(1);

for k=0:M-1
    accum=0;
    for m=0:k
        accum=accum+a(k+1,m+1)*rx(k-m+2);
    end
    gamma(k+2)=-accum/P;
    P=P*(1-abs(gamma(k+2))^2);
    a(k+2,1)=1;
    a(k+2,k+2)=gamma(k+2);
    for m=1:k
        a(k+2,m+1)=a(k+1,m+1)+gamma(k+2)*a(k+1,k-m+2);
    end
end
parameter=a(M+1,:);

bb=[1];
aa=a(M+1,:);

v=filter(aa,bb,var);
```

```
function  freq=fundfreq(frag)

% This function called fundfreq (stands for fundamental frequency)
% finds the fundamental frequency of the desired signal.
% for the K interval of a question using autocorrelation function.
% For a periodic signal with the period p, the autocorrelation function
% attains a maximum at 0,p,2p,..
% regardless of the time origin of the signal, the period can be estimated
% by finding the location the first maximum in the autocorrelation function.


%For using this function the user should enter the file segment fundfreq(frag).


Fs = 30;                          %Sampling frequency
K=length(frag);

y = xcorr(frag);                  % finds the autocorralation function

q = diff(abs(y(K:2*K-1)));        % differentiates the variable

z = q>0;                          % z = 1 if q is greater than 0

f = diff(z);                      %finds the indices where the 2nd derivative
                                  %is -1 or +1 which indicates peaks and valleys

peak = find(f<0);                 %finds the peak indices


m =K+peak;
[i,j]=max(abs(y(m)));             %finds the maximum peak value and its index

lofreq =find(f>=0);
   if length(lofreq)==length(f)
     freq=0;
   else
     freq = Fs/peak(j);
   end
```

```
function y=croscor(var1,var2)


% This function finds the cross correlation between two variables
% The first variable is prewhitened first by calling
% armod (stands for AR modeling) program.
% The function returns maximum and minimum of the croscorrelation
% and the lag that these maximum and minimum happen.
%To use this command the user must enter the two
%variable names to be correlated.
%
% eg.   croscor(variable1,variable2)

K=min(length(var1),length(var2));



M=10;                                          % Model order
v1=armod(var1,M);

yd= xcorr(v1(20:K),var2(20:K),'biased');

[maximum lagmax]=max(real(yd));

[minimum lagmin]=min(real(yd));

y=[maximum lagmax minimum lagmin];
```

```
function feature= feature(file_name,relevant,irrelevant,control,features,offset,CR_feature)

% This function produces a feature vector for a given file
% Relevent, irrelevent, and control are vectors which contain
% the questions these features are extracted from.
%
% eg. featurev(t79,[3 5],[1 4], [6 10],feature_list)

% The above example gives the features for
% the file t79 of the 3rd and 5th question which are relevent in this
% MGQT format, the 1st and 4th question which are irrelevent
% and the 6th and 10th questions which are control

% feature_list=['10mean(frag ) ';
%               '20curve(frag )';
%               '30area(frag ) '];


feature_list = features;



% The channels are ordered as follows:
% 1:GSR, 2:HiCardio, 3:LowCardio, 4:DerLowCardio, 5:LowResp, 6:UpResp


% This is a matrix of the time delay after asking a question to start of extracting
% the feature, and finish extracting the feature for each channel.

Times=[
        2, 14;
        3, 9 ;
        3, 18;
        1, 8 ;
        2, 18;
        2, 18];

% These are preprocessing functions.
Preprocess=[  'detgsr';
              'dethic';
              'detlc ';
              'dercd ';
              'detlr ';
              'detur '];
```

```
data=zeros(6,length(file_name(:,5)));
% Standardize and detrend the channels and derive new channels

for i=1:6,
        data(i,:)=eval([Preprocess(i,:),'(file_name)']);
end


marker = file_name(:,5);    % 0 begin test and end test
                            % 0 examiner begins asking question
                            % 1 examiner finishes asking question
                            % 2 subject begins response to question
                            % 9 does not mark an event

begin = find(marker == 0);     % finds indecies where marker = 0 (question begins)
begin=begin(2:length(begin)); % elliminates the marker at the beginning of the test

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++
% This for loop creates feature vectors for each relevant quesion
%
% eg x = [mean(gsr),std(gsr),area(gsr),mean(lr),std(lr),area(lr),etc.....
% curve length,amplitude of peaks,# of peaks]
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++

feature_count=1;

for i = 1:max(find(relevant~=0)),
        question=relevant(i);

        for j=1:length(feature_list(:,1))
                        channel_number=eval(feature_list(j,1));
                        second_channel=eval(feature_list(j,2));
                        st=begin(question)+30*Times(channel_number,1);
                        fn=begin(question)+30*Times(channel_number,2);
                        st2=begin(question)-30*Times(channel_number,2);
                        fn2=begin(question)-30*Times(channel_number,1);
                        fr=feature_list(j,3:length(feature_list(1,:)));
                        frag=data(channel_number,st:fn);
                        frag2 = data(channel_number,st2:fn2);
                        if second_channel ~= 0
```

```
                    st3=begin(question)+30*Times(second_channel,1);
                    fn3=begin(question)+30*Times(second_channel,2);
                    frag3 = data(second_channel,st3:fn3);
                end
                tempy=eval(fr);
                for m = 1:length(tempy)
                    x(feature_count) = tempy(m);
                    feature_count=feature_count+1;
                end
        end
end
%-----------------------------------------------------------------
% Irrelevant questions

feature_count=1;

for i = 1:(max(find(irrelevant~=0))-offset)
        question=irrelevant(i);
        for j=1:length(feature_list(:,1))
                channel_number=eval(feature_list(j,1));
                second_channel=eval(feature_list(j,2));
                st=begin(question)+30*Times(channel_number,1);
                fn=begin(question)+30*Times(channel_number,2);
                st2=begin(question)-30*Times(channel_number,2);
                fn2=begin(question)-30*Times(channel_number,1);
                fr=feature_list(j,3:length(feature_list(1,:)));
                frag=data(channel_number,st:fn);
                frag2 = data(channel_number,st2:fn2);
                if second_channel ~= 0
                    st3=begin(question)+30*Times(second_channel,1);
                    fn3=begin(question)+30*Times(second_channel,2);
                    frag3 = data(second_channel,st3:fn3);
                end
                tempy=eval(fr);
                for m = 1:length(tempy)
                    y(feature_count) = tempy(m);
                    feature_count=feature_count+1;
                end
        end
end
```

```
%-----------------------------------------------------------------
% Control questions

feature_count=1;

for i = 1:max(find(control~=0)),
        question=control(i);

        for j=1:length(feature_list(:,1))
                channel_number=eval(feature_list(j,1));
                second_channel=eval(feature_list(j,2));
                st=begin(question)+30*Times(channel_number,1);
                fn=begin(question)+30*Times(channel_number,2);
                st2=begin(question)-30*Times(channel_number,2);
                fn2=begin(question)-30*Times(channel_number,1);
                fr=feature_list(j,3:length(feature_list(1,:)));
                frag=data(channel_number,st:fn);
                frag2 = data(channel_number,st2:fn2);
                if second_channel ~= 0
                        st3=begin(question)+30*Times(second_channel,1);
                        fn3=begin(question)+30*Times(second_channel,2);
                        frag3 = data(second_channel,st3:fn3);
                end
                tempy=eval(fr);
                for m = 1:length(tempy)
                        z(feature_count) = tempy(m);
                        feature_count=feature_count+1;
                end
        end
end
%-----------------------------------------------------------------------------

% control & relevant


feature_count=1;

for i = 1:max(find(relevant~=0)),
                for k=1:max(find(control~=0)),
                        q(k)=abs(relevant(i)-control(k));
                end

                [a b]=min(q);
```

```
                question1=relevant(i);
                question2=control(b);

                for j=1:length(CR_feature(:,1))
                        channel_number=eval(CR_feature(j,1));
                        st=begin(question1)+30*Times(channel_number,1);
                        fn=begin(question1)+30*Times(channel_number,2);
                        st2=begin(question2)+30*Times(channel_number,1);
                        fn2=begin(question2)+30*Times(channel_number,2);
                        fr=CR_feature(j,3:length(CR_feature(1,:)));
                        frag1=data(channel_number,st:fn);
                        frag2=data(channel_number,st2:fn2);
                        tempy=eval(fr);
                        for m = 1:length(tempy)
                                w(feature_count) = tempy(m);
                                feature_count=feature_count+1;
                        end
                end
end

feature=[x,y,z,w]';
```

```
function  isd_dif=isd(frag1,frag2)

% This is a integrated  spectral difference(isd) function that finds the cumulativespectral
% density of a control-relevant pair, then calculates the difference between the
% isd of control and the relevant for a part of a question.
% This function returns the max or min and the frequency (points)
% where this max or min happens and the area underneath  this difference.

% To use this command the user must enter the two variable names.
% The first variable is a control question fragment and the second is
% a relevant question fragment.
% eg.  isd1(variable1,variable2)

Fs = 30;
K=min(length(frag1),length(frag2));

nnp =1;
np = 2^nnp;
L = K/np;
L=2^(nextpow2(L));

M= spectrum (frag1,L);        %spectral density of the first (control) question
N= spectrum (frag2,L);        %spectral density of the second(relevant) question

pqc = cumsum(M(:,1));         %Cumulative sum of the integrated spectral density
pqr = cumsum(N(:,1));         %Cumulative sum of the integrated spectral density

clear M
clear N
hc = pqc/pqc(L/2);
hr = pqr/pqr(L/2);

CR_dif= hr' - hc';
 if (abs(max(CR_dif))>abs(min(CR_dif)))
    [CR_dif, mpoint]=max(CR_dif);
 else
    [CR_dif ,mpoint]=min(CR_dif);
 end
isd_dif=[ CR_dif  mpoint trapz(hr'-hc')];
```

```
feature_list=[   '10mean(frag)          ';
                 '10curve(frag)         ';
                 '10area(frag)          ';
                 '10med_dif(frag,8)     ';
                 '10max_min(frag)       ';
                 '10max(frag)           ';
                 '10min(frag)           ';
                 '10mdif(frag)          ';
                 '20mean(frag)          ';
                 '20curve(frag)         ';
                 '20area(frag)          ';
                 '20ampcard(frag)       ';
                 '20dampcard(frag)      ';
                 '20peaknumc(frag)      ';
                 '20med_dif(frag,5)     ';
                 '20max_min(frag)       ';
                 '20max(frag)           ';
                 '30min(frag)           ';
                 '20min(frag)           ';
                 '20mdif(frag)          ';
                 '20minampc(frag)       ';
                 '30mean(frag)          ';
                 '30curve(frag)         ';
                 '30area(frag)          ';
                 '30med_dif(frag,5)     ';
                 '30max_min(frag)       ';
                 '30max(frag)           ';
                 '30mdif(frag)          ';
                 '40mean(frag)          ';
                 '40min(frag)           ';
                 '40mdif(frag)          ';
                 '40curve(frag)         ';
                 '40area(frag)          ';
                 '40med_dif(frag,5)     ';
                 '40max_min(frag)       ';
                 '40max(frag)           ';
                 '50mean(frag)          ';
                 '50curve(frag)         ';
                 '50area(frag)          ';
                 '50ampr(frag)          ';
                 '50peaknumr(frag)      ';
                 '50ie(frag)            ';
                 '50dampr(frag)         ';
                 '50ieie(frag, frag2)   ';
                 '50med_dif(frag,8)     ';
                 '50max_min(frag)       ';
                 '50max(frag)           ';
                 '50min(frag)           ';
                 '50mdif(frag)          ';
                 '50minampr(f   ;)      ';
                 '60mean(frag)          ';
```

```
                    '60curve(frag)          ';
                    '60area(frag)           ';
                    '60ampr(frag)           ';
                    '60dampr(frag)          ';
                    '60peaknumr(frag)       ';
                    '60ie(frag)             ';
                    '60ieie(frag, frag2)    ';
                    '60med_dif(frag,8)      ';
                    '60max_min(frag)        ';
                    '60max(frag)            ';
                    '60min(frag)            ';
                    '60mdif(frag)           ';
                    '60minampr(frag)        ';
                    '10std(frag)            ';
                    '20std(frag)            ';
                    '30std(frag)            ';
                    '40std(frag)            ';
                    '50std(frag)            ';
                    '60std(frag)            ';
                    '20armod1(frag)         ';
                    '20cor1(frag)           ';
                    '50cor1(frag)           ';
                    '26croscor(frag,frag3)  ';
                    '26psdcoh1(frag,frag3)  '];

CR_feature=[
                    '10isd1(frag1,frag2)        ';
                    '20isd1(frag1,frag2)        '];


lf=length(feature_list(:,1));
cd \mgqt\g1
files1
for d=1:3
        if d==2
                cd \mgqt\g2
                files2
        elseif d==3
                cd \mgqt\non_dec
                filesn
        end

   for k=1:length(flist(:,1))
        file_name=[flist(k,:)];
        flength=length(file_name);
        question=['ZZ',num2str(file_name(3:flength-1)),'4'];
% creates the name of the file that holds the questions(zz*.014) .
```

```
        eval(['load ', file_name]);                % load the data & the file with the
        eval(['load ', question]);                 % question number
        file_name=file_name(1:flength-4);          %eleminates the extention(.013)
        question=question(1:flength-4) ;           % in order to use the data.
        Q=eval(question);
        l_rel=max(find(Q(2,:)~=0));                %The length of relevant questions
        l_con=max(find(Q(4,:)~=0));                %The length of control questions
        l_irr=max(find(Q(3,:)~=0));                %The length of irrelevant questions
        qover =l_con+l_rel+l_irr-10;               % finds the number of questions over 10
        offset=qover*(qover>0);
        CRlength=l_rel*6;
        size_M=(10+(qover<0)*qover)*(lf+18)+CRlength;  %total size of features

        initial=zeros(10*(18+lf)+30,1);            %Initializing M with a 10*lf zeros
        M(:,k)=initial;
        M(1:size_M,k)=feature(eval(file_name),[Q(2,:)],[Q(3,:)],[Q(4,:)],feature_list,offset,C
        R_feature);

        eval(['clear ',upper(file_name)])
        eval(['clear ',upper(question)])

    end

        save new_feat M lf flist
        clear M
end
```

```
clear
featlength=23;
load new_feat
for k=1:length(flist(:,1))
        file_name=[flist(k,:)];
        flength=length(file_name);
        question=['ZZ',num2str(file_name(3:flength-1)),'4'];
        eval(['load ',question]);              % load the file with the question numbers.
        Q=eval(question(1:flength-4));         % in order to use the data.
        l_rel=max(find(Q(2,:)~=0));            %The length of relevant questions
        l_con=max(find(Q(4,:)~=0));            %The length of control questions
        l_irr=max(find(Q(3,:)~=0));            %The length of irrelevant questions


% Averaging relevant questions
        for j=1:lf-5+featlength
                m=(j-1)*7;
                clear r
                for i=1:l_rel
                    r(i)=M((i-1)*(lf-5+featlength)+j,k);    %finds the feature values
                end                                         %for all the relevant questions.

                feat_vec(m+1,k)=mean(r);                    %returns mean value for relevant
                feat_vec(m+2,k)=mean(r);
                feat_vec(m+3,k)=max(r);
                feat_vec(m+4,k)=min(r);
                feat_vec(m+5,k)=max(r);
                feat_vec(m+6,k)=min(r);
                feat_vec(m+7,k)=max(r);
        end
  qover =l_con+l_rel+l_irr-10 ;                     %The number of questions over 10
  offset=qover*(qover>0);
  l=(l_irr-offset+l_rel)*(lf-5+featlength);         %The position of the
  cr_l=l+l_con*(lf-5+featlength);                   %first control question

%-----------------------------------

% Averaging control questions

  for j=1:lf-5+featlength
        clear c
        m=(j-1)*7;
        for i=1:l_con
                c(i)=M((i-1)*(lf-5+featlength)+j+l,k);        %finds the feature values for
```

```
        end                                          %all the control questions.

%feature values for control questions

        f(m+1,k)=feat_vec(m+1,k)-mean(c);
                if (feat_vec(m+2,k)+mean(c)==0)
                        f(m+2,k)=100;
                else
                        f(m+2,k)=2*(feat_vec(m+2,k)-
                         mean(c))/(feat_vec(m+2,k)+mean(c));   %for every feature.
                end
        f(m+3,k)=feat_vec(m+3,k)-max(c);
        f(m+4,k)=feat_vec(m+4,k)-min(c);
        f(m+5,k)=feat_vec(m+5,k)-min(c);
        f(m+6,k)=feat_vec(m+6,k)-max(c);
                if max(c)==0
                        f(m+7,k)=100;
                else
                        f(m+7,k)=feat_vec(m+7,k)/max(c);
                end
        end

%-------------------------------------------------
% feature values for control_relevant

  for j=1:6
        m=(j-1)*3;
        clear cr
        for i=1:l_rel
                cr(i)=M((i-1)*6+j+cr_l,k);
        end

        f(m+1+(lf-5+featlength)*7,k)=mean(cr);
        f(m+2+(lf-5+featlength)*7,k)=max(cr);
        f(m+3+(lf-5+featlength)*7,k)=min(cr);
  end

decep(1,k)=Q(1:1);                  % finds if file is deceptive or not
                                    % creates 1 if deceptive and 0 if not.
eval(['clear ',upper(question(1:flength-4))]);
end


save fn_dec f decep
```

Time Domain Features For The Fuzzy Set Classification Of Polygraph Data

EE 297

Dr. Benjamin Knapp
Electrical Engineering Department
San Jose State University

Eric Jacobs
November 1993

Abstract

A polygraph examination is the most popular method used to determine if an individual is being truthful or deceptive. During an examination, a subject is asked a series of questions and the physiological responses to the questions are recorded using a polygraph. The three physical responses currently obtained from a polygraph examinations are blood pressure, respiration, and skin conductivity. Polygraph charts are usually analyzed by a human interpreter for evidence of truth or deception; however, computer algorithms are now being used to verify results [1][2].

In this project, the K nearest neighbor algorithm was used to determine truth or deception. By using this adaptive fuzzy system, it was possible for the computer evaluation of the polygraph to adapt to individual differences in the physiological responses. Two algorithms were necessary for this project. The first was a parsing algorithm which preprocessed polygraph data and extracted features from it. These features can be separated into three domains: time domain, frequency domain, and correlation domain. The second was the K nearest neighbor fuzzy classifier which analyzed the data from the parsing algorithm and determined the possibility of deception.

Contents

1.1 History	2
1.2 Modern Test Formats	2
1.3 Present Day Equipment	3
2.1 Fuzzy Set Theory	5
3.1 MGQT	8
4.1 File Formats	8
5.1 Preprocessing	10
5.2 Time Domain Feature Extraction	15
5.3 Feature Extraction Methods	17
6.1 Conclusion	17
References	18
Appendices	20

1.1 History

The first attempt to use a scientific instrument in an effort to detect deception occurred around 1895 [3]. That was the year that Cesar Lombroso published the results of his experiments in which a hydrosphygmograph was used to measure the blood pressure-pulse changes of criminals in order to determine whether or not they were deceptive. Although the hydrosphygmograph was originally intended to be used for medical purposes, Lombroso found that it worked well for lie detection. Lombroso may have been the first to use a peak of tension test format. This was done by showing a suspect a series of photographs of children, one being the victim of sexual assault. If the suspect did not react more to the victims picture than the pictures of the other children, Lombroso concluded that the suspect did not know what the victim looked like and therefore was not the alleged perpetrator.

In 1914 Vittorio Benussi published his research on predicting deception by measuring recorded respiration tracings [4]. He found that if the length of inspiration were divide by the length of expiration, the ratio would be larger after lying than before lying and also before telling the truth than after telling the truth. In 1921 John A. Larson constructed an instrument capable of simultaneously recording blood pressure pulse and respiration during an examination [3][4]. Larson reported accurate results which prompted Leonarde Keeler to construct a better version of this instrument in 1926 [3][4].

The use of galvanic skin response in lie detection began during the turn of the century. It's usefulness, however, did not become evident until the 1930's during which time several articles written by Father Walter G. Summers of Fordham University in New York [4]. In these articles he reports over 90 criminal cases in which examination using the galvanic skin response had all been successful and confirmed by confession or supplementary evidence. The usefulness of the galvanic skin response prompted Keeler to add an galvanometer to his polygraph. At the time of Keelers death in 1949, the Keeler Polygraph recorded blood pressure-pulse, respiration, and galvanic skin response [3].

1.2 Modern Test Formats

The effectiveness of a polygraph examination is often the result of the test format that is used. A polygraph test format consists of an ordered combination of relevant questions about an issue, control questions that provide a physical response for comparison, and irrelevant questions that also provide a response or the lack of a response for comparison [1][4]. Three general types of test formats are in use today. These are Control Question Tests, Relevant-Irrelevant Tests, and Concealed Knowledge Tests. Each of the general test formats may have a number of more specific variations. Each test consists of two to five charts containing a prescribed series of questions. The test format that is used in an examination is determined by the test objective [3][4].

The concealed knowledge test, also called peak of tension test, is used when facts about a crime are known only by the investigators and not by the public. In this case, a subject would not know the facts unless he or she was guilty of the crime. For example, if a gun was used in a crime and the public did not know the caliber, an examiner could ask a suspect if it was a 22 caliber , a 38 caliber, or a 9mm. If the gun used was a 9mm

and the suspect was deceptive, a polygraph chart would probably indicate evidence of deception.

A control question test is often used in criminal investigations. In this type of test a series of relevant, irrelevant, and control questions are asked. A relevant question is one which is specific to the crime being investigated. For example, " Did you molest the child?". A control question is designed to make the subject feel uncomfortable. It is not specific to the crime being investigated however it may be related in an indirect way. A control question that could follow the relevant question stated above is "Have you ever forced yourself on another person sexually ?". The control questions are compared to the relevant questions and if the responses to the relevant questions are greater, the subject is usually classified as deceptive. Irrelevant questions are used as buffers. Examples of irrelevant questions are "Are the lights in this room on?" or "Is today Monday?".

Relevant-Irrelevant tests are usually used to test people trying to obtain security clearance or get a job. In this test, relevant questions are compared to irrelevant questions. Very few control questions are asked. The purpose of control questions in this test is to make sure that the subject is capable of reacting at all.

1.3 Present Day Equipment

The most popular polygraph machines today are the Reid Polygraph developed in 1945 and the Axciton Systems computerized polygraph developed in 1989 [1][11]. The Reid polygraph scrolls a piece of paper under pens that record the biological signals. The Axciton polygraph digitizes physiological signals and uses a computer to process them. The sampling frequency of the Axciton machine is 30 Hz. Axciton provides a computer based system for ranking the subject responses but allows printouts of the charts to be scored by hand the traditional way. The Axciton and Reid polygraphs are shown in figures 1 and 2 respectively.

Both machines record the same biological signals using standard methods. Blood pressure is measured by placing a standard blood pressure cuff on the arm over the brachial artery. Respiration is monitored by placing rubber tubes around the abdominal area and the chest of the subject. This results in two signals, an upper and lower respiratory signal. Skin conductivity is measured by placing electrodes on two fingers of the same hand.

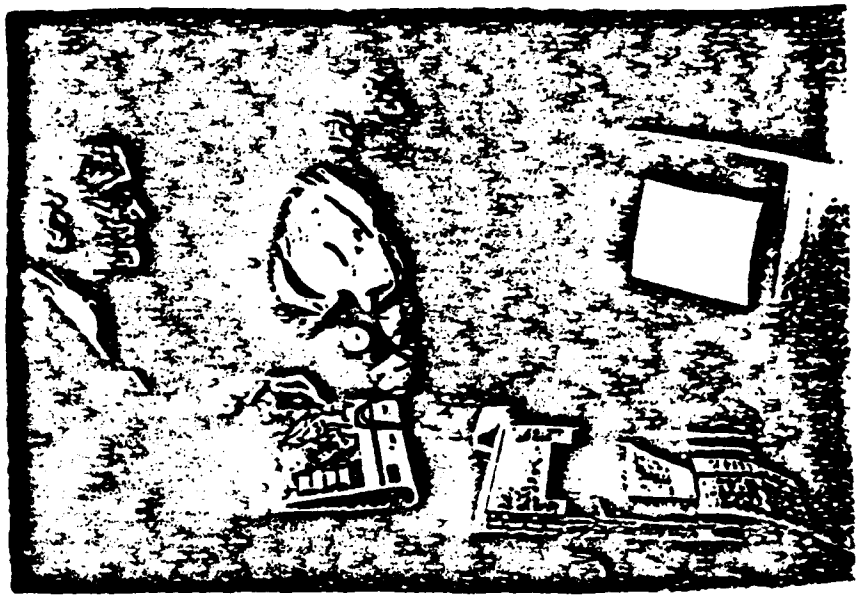


Figure 1 Axciton Polygraph [1]

Figure 2 Reid Polygraph [3]

2.1 Fuzzy Set Theory

In 1965 fuzzy sets were introduced by Lofti Zadeh [5][6]. They provided a new way to represent vagueness and made description of many situations much easier. For example, it is not practical to say that all temperatures below 72 degrees Farenheit are cold and all temperatures above are hot. Instead, temperatures between 50 and 72 would by described as cool, temperatures between 30 and 50 would be considered cold, and anything below 30 would be very cold. One way to describe this situation is through the use of fuzzy set theory. In fuzzy set theory an element is not defined as belonging or not belonging to a given set. Instead, it has a degree of membership in a set which is characterized by a compatibility function uA [6] [7]. The compatibility function, also called a membership function, states the degree of membership in a set "A" and has a range [0,1]. An illustration of how this applies to the temperature example above is illustrated in figure 1 and described below.

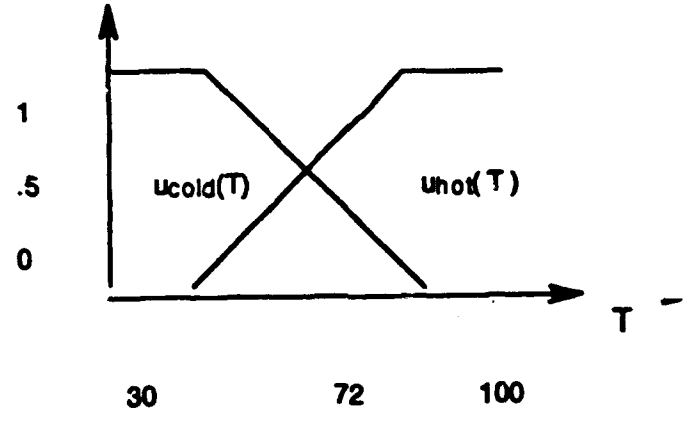


Figure 3 Compatibility functions ucold(T) and uhot(T) vs. temperature.

Here, $u_{cold}(T)$ and $u_{hot}(T)$ are the degrees of membership in each set and T is the temperature in Farenheit. Figure 1 shows that the temperatures around 72 degrees have membership in $u_{cold}(T)$ and $u_{hot}(T)$. These memberships have values around .5 which represents cool or warm. As the cooler temperatures decrease, $u_{cold}(T)$ increases thus representing a colder situation. Once the temperatures become less than 30 degrees, $u_{cold}(T)$ obtains a membership value of 1 which indicates very cold temperatures.

Fuzzy set theory is often thought of as another form of probability theory. In actuality, the two are very different [8]. In Bayesian probability theory, elements either belong or do not belong to a given set, and a probability density function determines the likelihood. For example, a light may be either on or off and the probability of either event occurring will depend on some statistical parameters (Is the room occupied? Is it dark out? etc.). The following is an example of the difference between fuzzy logic and Bayesian probability theory [6].

Example 1

Let L = set of all liquids, and let fuzzy subset l = {all (potable) liquids}. Suppose you had been in the desert for a week without drink and you came upon two bottles marked "C" and "A" as in figure 4a.

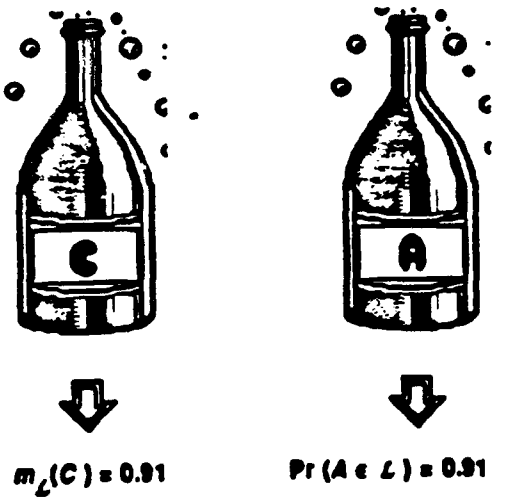


Figure 4a Liquids before observation

Confronted with this pair of bottles, and given that you must drink from the one that you choose, which would you choose to drink from? Most readers, when presented with this experiment, immediately see that while "C" could contain, say, swamp water, it would not (discounting the possibility of a Machiavellian fuzzy modeler) contain liquids such as hydrochloric acid. That is, membership of 0.91 means that the contents of "C" are fairly similar to perfectly potable liquids (e.g., pure water). On the other hand, the probability that " A" is potable = 0.91 means that over a long run of experiments, the contents of A are expected to be potable in about 91% of the trials; in the other 9% the contents will be deadly - about 1 chance in 10. Thus, most subjects will opt for a chance to drink swamp water.

There is another facet to this example, and it concerns the idea of observationion. Continuing then, suppose that we examine the contents of "C" and "A" and discover them to be as shown in figure 4b. Note that, after observation, the membership value for "C" is unchanged while the probability value for A drops from 0.91 to 0.0.

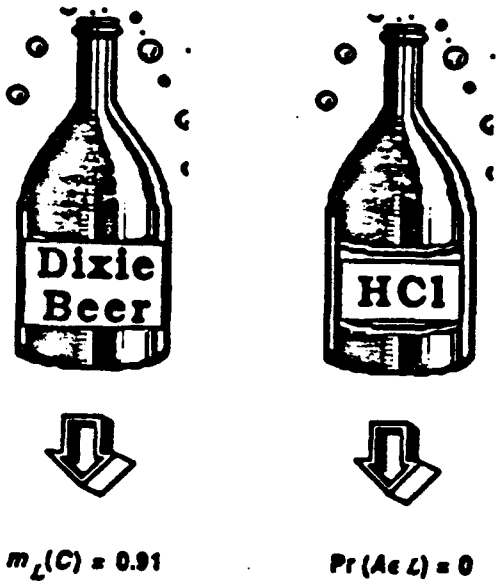


Figure 4b Liquids after observation

This example shows that these two models possess philosophically different kinds of information: fuzzy memberships, which represent similarities of objects to imprecisely defined properties; and probabilities, which convey information about relative frequencies.

3.1 MGQT

The test format used in this project was the MGQT test format. It is a type of control question test in which relevant, irrelevant, and control questions are asked in the order given in table 1 [9][12]. Before each test, the questions that will be asked are discussed with the subject. The series of questions is asked three times in the order specified in table 1. This produces three test charts. The examiner waits about 20 seconds between each question.

Not all of the Axciton charts used in this study follow the format of table 1 exactly. Many examiners rearranged the order in which the questions were asked. All polygraph charts used, however, were variations of this test. For example, one examiner used a test format in which questions 3 and 4 were switched. Many of the examiners changed the order in which the questions were asked in the second and third charts.

Question	Type of Question
1	irrelevant
2	irrelevant
3	relevant
4	irrelevant
5	relevant
6	control
7	irrelevant
8	relevant
9	relevant
10	control

Table 1 MGQT question format

4.1 File Formats

Axciton files, digitized polygraph data from the axciton polygraph, were obtained from the National Security Agency (NSA) in standard MSDOS format. The sampling frequency of the data was 30Hz. Each test consisted of nine files. The labling of the files is shown in table 2 and the purpose of each file is explained below.

Chart 1	Chart 2	Chart 3
$$xxxxxx.011	$$xxxxxx.021	$$xxxxxx.031
$$xxxxxx.012	$$xxxxxx.022	$$xxxxxx.032
$$xxxxxx.013	$$xxxxxx.023	$$xxxxxx.033

Table 2 File format

As stated in the section above, each examination is composed of three charts. The chart number is specified by the second number after the period. The third number after the period represents the type of file.

$$xxxxxx.0x1 is the event marker file which contains the length of the chart and the event markers. The start and end of an examiners question is marked with a 0 and 1, respectively. The beginning of the subjects response is indicated with a 2 and the rest of the file is marked with 9's. File $$xxxxxx.0x2 is the file containing the biological signals. These signals correspond to the marker file. File $$xxxxxx.0x3 contains the questions and labels them relevant, irrelevant, or control.

An ASCII file of five columns is created by using $$xxxxxx.0x1 and $$xxxxxx.0x2 and a program provided by the NSA. An example of this file along with a description of the function of each file is shown in table 3 [12].

	Event Marker	**FileChart Data**	**FileQuestion TextFile**
	$$xxxxxx.0x1	$$xxxxxx.0x2	$$xxxxxx.0x3
Axciton File	Contains the length of the chart, the number of channels, and the position of the event marker.	Contains the digitized series values formatted according to flags in the Event Marker File.	Contains the script of of questions or a shorthand script of questions.
Processing Notes	Becomes the 5th column of ASCII file. 0=start of a question 1=end of a question 2=start of response 9=No Event Marker	Becomes 1st-4th columns of ASCII file. Column 1-GSR Column 2-Cardio Column 3-Upper Resp Column 4-Lower Resp	Files used to determine deviations from standard test format.

ASCII File Format (with column labels)

	File Row	GSR	Cardio	UR	LR	EvMark
DOS File	1	1983	1931	1482	1083	9
	2	1983	1922	1483	1084	9
	3	1983	1913	1483	1084	9
	4	1983	1906	1483	1085	9

Table 3 File description and example

5.1 Preprocessing

MATLAB was used to display the signals and implement all of the filters and feature extraction algorithms. First, the four biological signals were processed into six channels. Hamming windowed FIR filters were used to create these channels and eliminate noise. A low frequency cardiovascular channel was produced by lowpass filtering the cardiovascular signal at .5 Hz using a 134 tap lowpass filter. Then, a high frequency cardiovascular channel was produced by highpass filtering the cardiovascular signal at .5 Hz using a 134 tap highpass filter. The derivative of the low frequency channel was then used to create a third channel. To eliminate noise, the upper and lower respiratory signals were lowpass filtered at 1.2 Hz using a 160 tap filter. Noise was eliminated from the galvanic skin response by using a 100 tap lowpass filter with a cutoff frequency of .5 Hz. Any DC trends that existed within a chart were eliminated using the detrend function in MATLAB. This function finds the best straight line fit to the data and then subtracts the line from the data. Each signal was normalized by dividing by its standard deviation. The raw data and results of this processing are shown in figures 5-14.

Fragments of each signal were accessed before features were extracted. These fragments were successfully used by Brian M. Duston of the Naval Control and Ocean Surveillance Center in his study and are given in table 4 [9]. The start and end points given in table 4 refer to the time elapsed after the question was asked by the examiner.

Channel	Start	End
GSR	2 sec.	14 sec.
Upper respiratory	2 sec.	18 sec.
Lower respiratory	2 sec.	18 sec.
Low frequency cardiovascular	2 sec.	18 sec.
High frequency cardiovascular	3 sec.	9 sec.
Derivative of low frequency cardiovascular	0 sec.	8 sec.

Table 4 Time fragments used in feature extraction

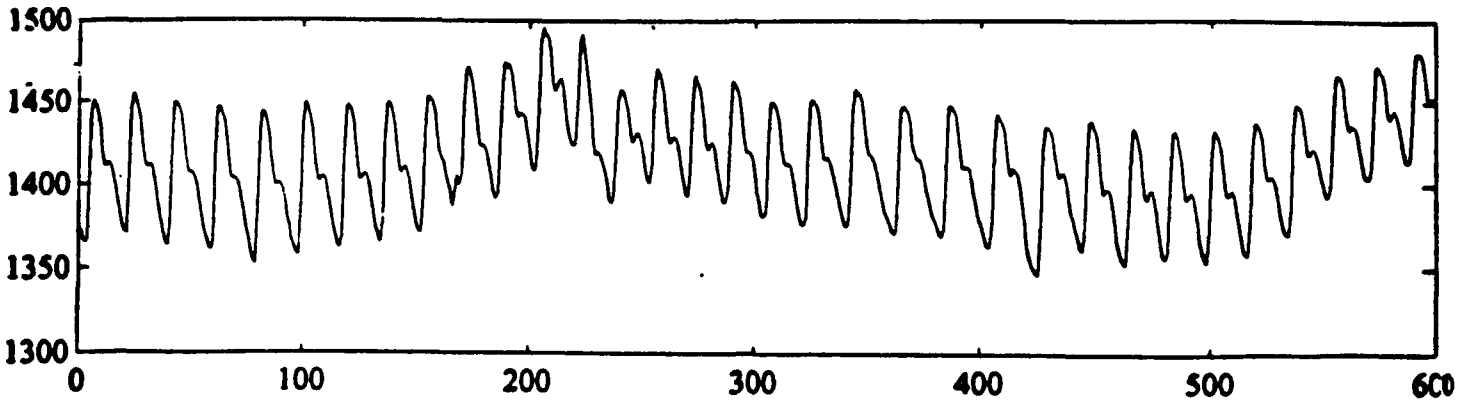


Figure 5 Cardiovascular

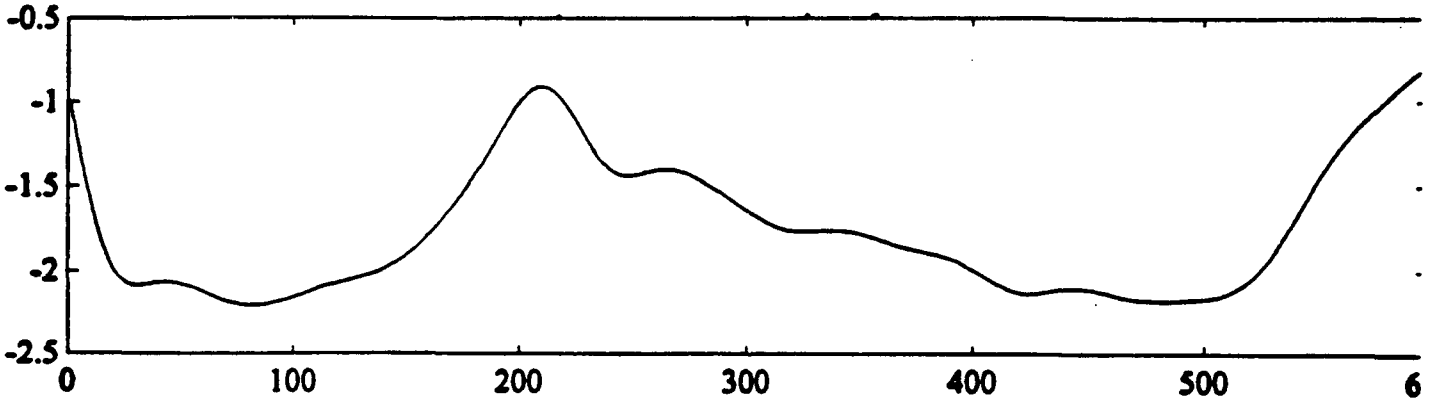


Figure 6 Preprocessed Low Frequency Cardiovascular

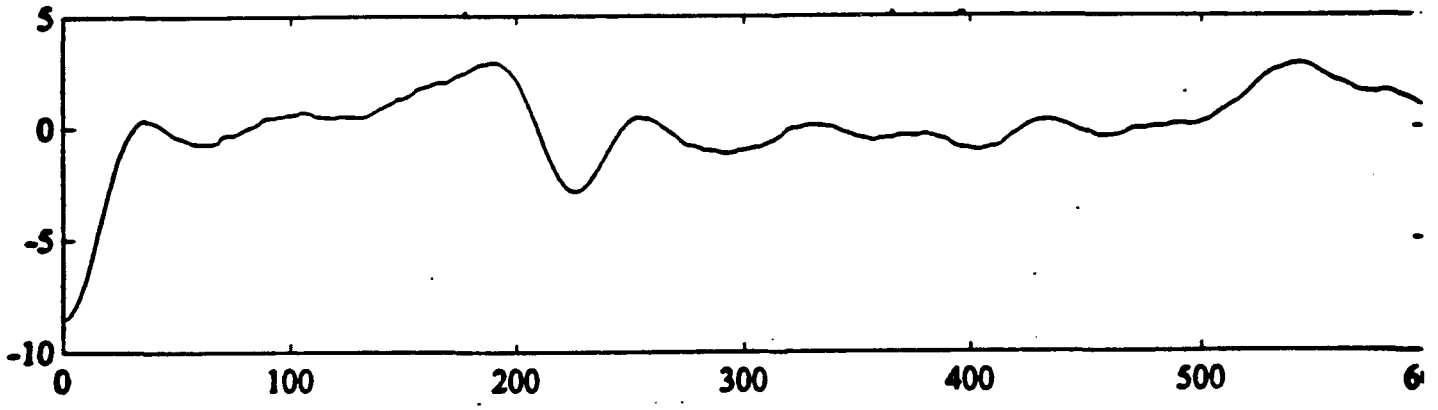


Figure 7 Preprocessed Derivative of Low Frequency Cardiovascular

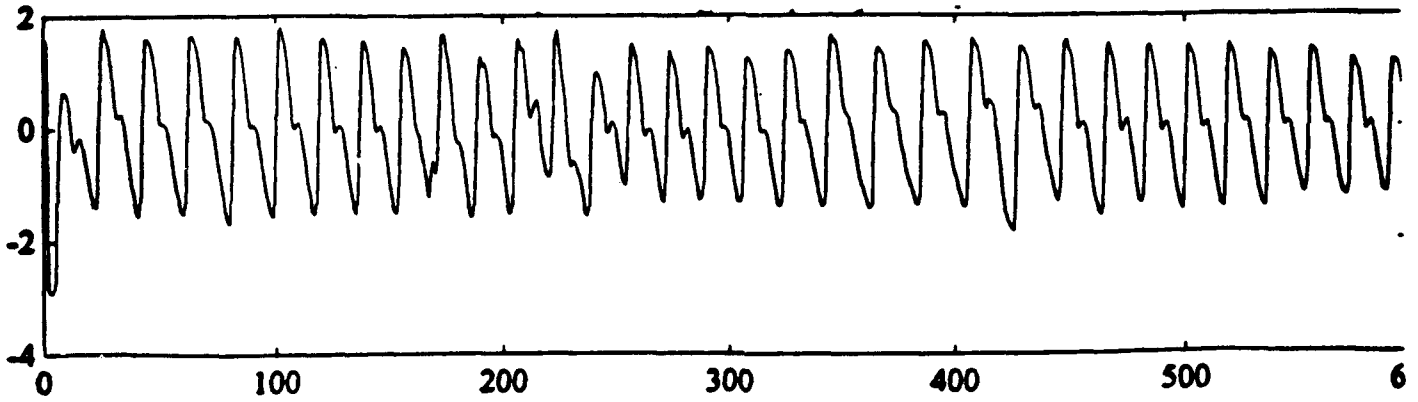


Figure 8 Preprocessed High Frequency Cardiovascular

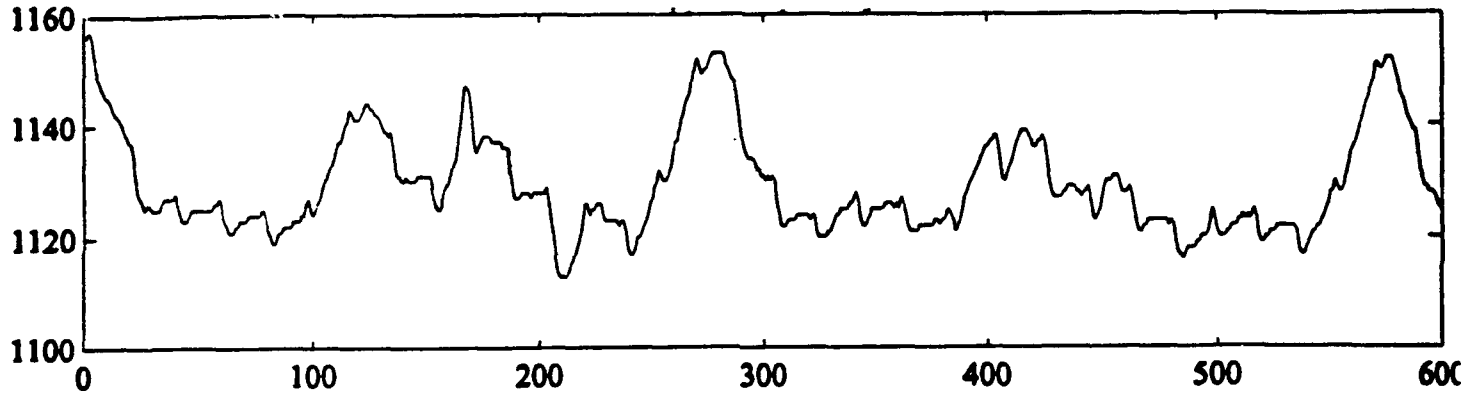


Figure 9 Upper Respiratory

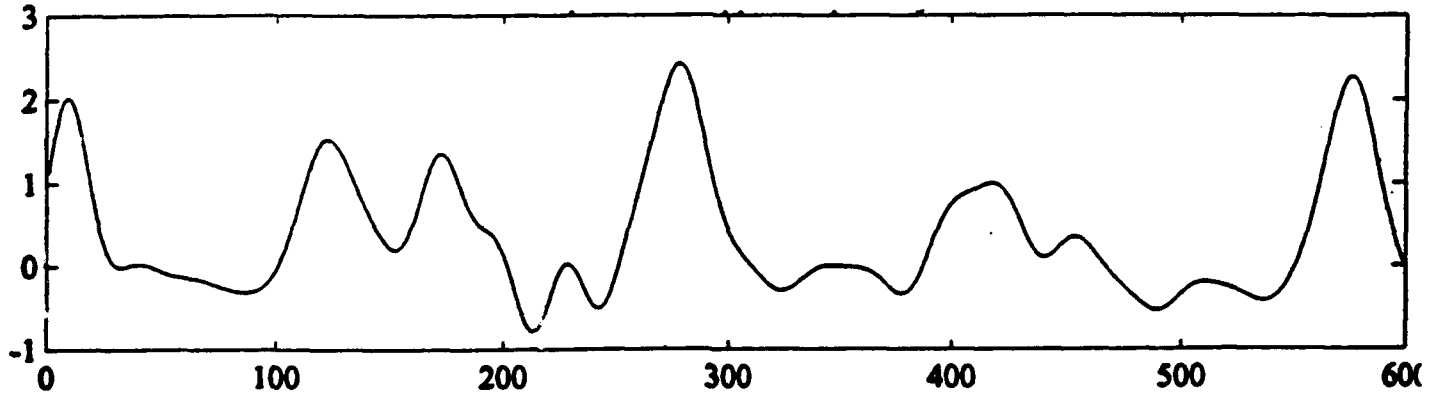


Figure 10 Preprocessed Upper Respiratory

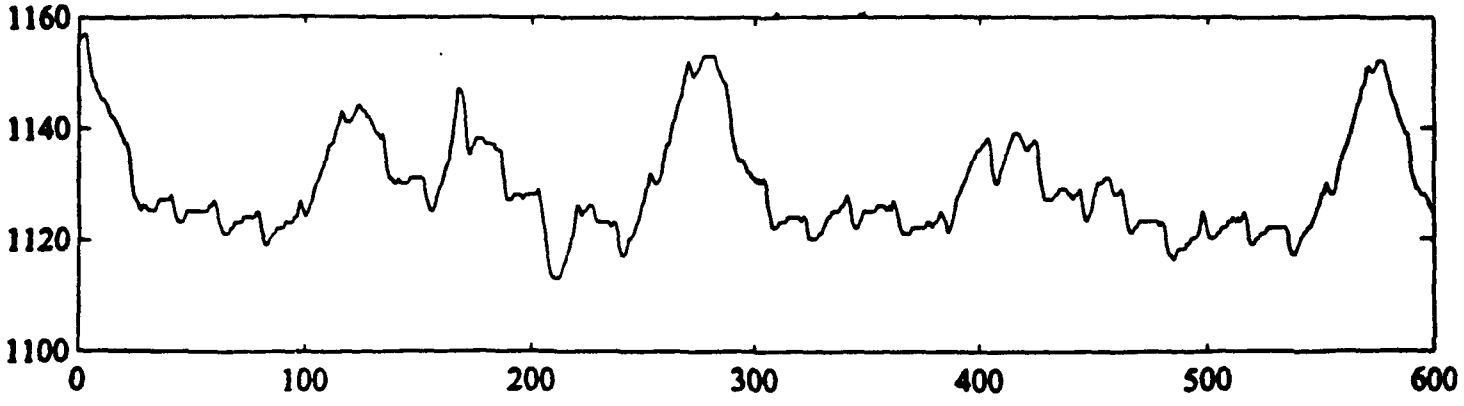


Figure 11 Lower Respiratory

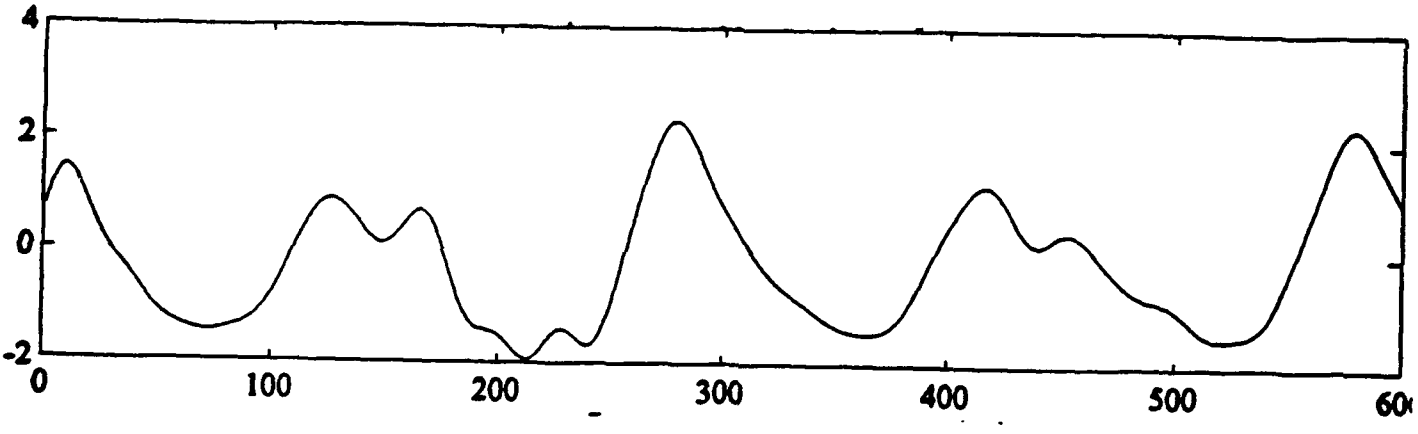


Figure 12 Preprocessed Lower Respiratory

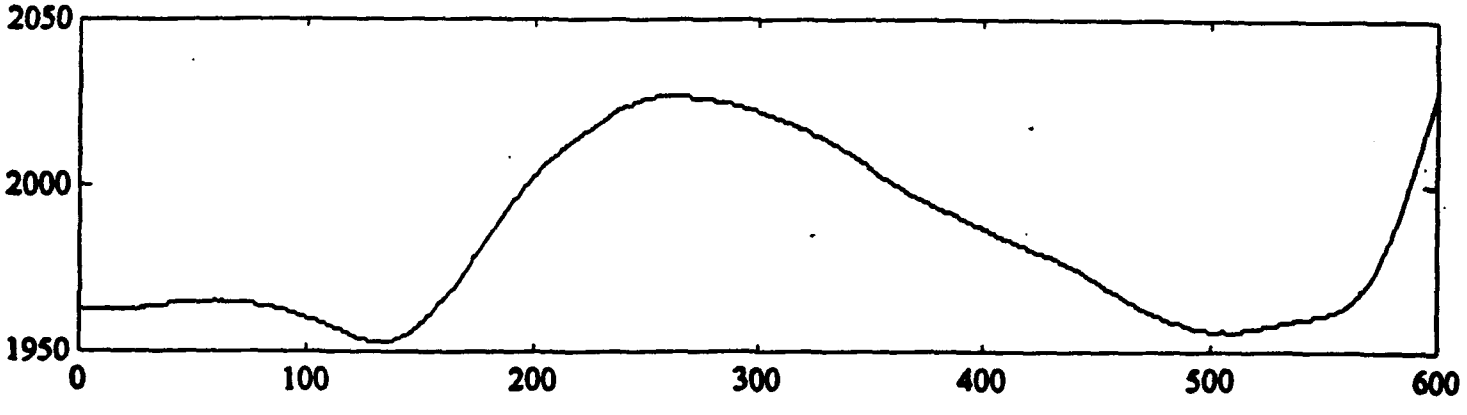


Figure 13 GSR

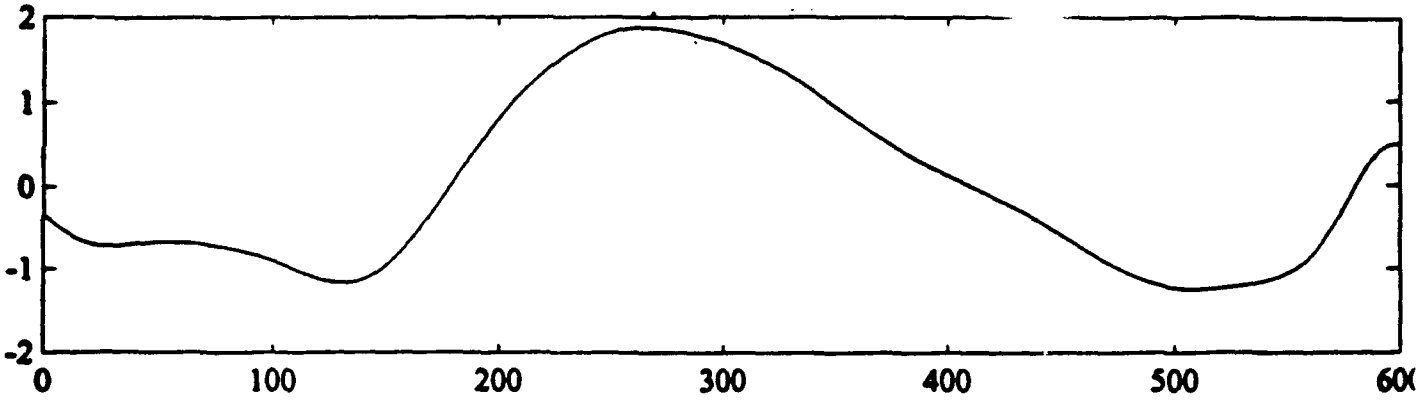


Figure 14 Preprocessed GSR

5.2 Time Domain Feature Extraction

Many of the time domain features were chosen by talking to examiners and finding out what was important to them in an examination [10][11]. One feature examiners use to determine deception involves the height of the peaks in the respiratory signal. If the peaks become smaller or staircase during a relevant question there is a good chance that the subject is being deceptive. From looking at different polygraph charts it could be seen that individual reactions may vary slightly with time. For this reason, many features were extracted from the respiratory channels in order to determine if the deceptive characteristics described above may be present. One feature extracted from the respiratory signal was the average height of the peaks. Because the time fragments from which the features are extracted remain constant, this feature may not give good results for subjects reacting early or late. For this reason, the minimum peak height was also used as a feature.

To try and capture the effect of staircasing, the average of the derivative of the amplitudes of the peaks was used as feature. To compensate for early and late reactions, the maximum of the derivative of the amplitudes of the peaks was also used as a feature.

Another respiratory feature used in this project was the curve length. This feature was successfully used and researched by Howard Timm in the early 1980's[10][13]. Interest in curve length lead to curiosity about the area under the respiratory curve. For this reason it was also extracted to see if it could be used as a feature. Because people tend to breath quicker when they are stressed or nervous, the number of peaks produced during a given period of time was used as a feature.

Because it was one of the first features used to successfully determine deception, Benussi's I/E ratio was tested [3][4]. Benussi's method requires that the I/E ratio of the subject is calculated before and after the examiner asks a question. The value of the I/E ratio calulated after the question is asked is then divided by the value of the I/E ratio before the question is asked. According to Benussi's findings, if the ratio is greater than one, the subject is deceptive. In an attempt to reduce the number of computations required for Benussi's method, a modification of Benussi's feature was tested. In the modification of Benussi's test, the ratio was taken only after the question was asked and was not compared to the subjects I/E ratio before the question was asked.

The examiners we spoke to would usually try to find evidence of deception in respiratory signals first. If a subject did not show a strong respiratory response however, the examiner would analyze the subjects cardiovascular response. Because a subjects heart rate will often increase when deceptive, the number of peaks in the high frequency cardiovascular signal was used as a feature. From looking at many charts, it became evident that some of the processing used in extracting features from the respiratory channels would also be useful in determining deception from the high frequency cardiovascular channel. For this reason, the average of the peak height, minimum of the peak height and curve length were extracted from the high frequency cardiovascular channel in order to determine if they would be useful features.

Many of the standard statistcal features used in other computerized polygraph algorithms were also examined [9]. These features included the mean, the standard deviation, the maximum amplitude, and the minimum amplitude of the signal. Variations

of these such as the minimum subtracted from the maximum were also examined. Although the original use of the curve length and area was to determine deception from the respiratory channel, it was extracted from the GSR and cardiovascular channels as well. It was not possible from looking at the signals to determine if the curve length had changed, but almost any change in a signal would affect this feature. A list of the features extracted from each channel are given in table 5. The programs used to extract these features were written in MATLAB and are included in the appendix of this report.

High frequency cardiovascular

1) mean of signal
2) standard deviation of signal
3) minimum value of signal
4) maximum value of signal
5) curve length of signal
6) area under signal
7) average amplitude of peaks
8) minimum amplitude of peaks
9) derivative of the amplitudes of the peaks in the signal
10) number of peaks in the signal
11) minimum subtracted from maximum

GSR

1) mean of signal
2) standard deviation of signal
3) minimum value of signal
4) maximum value of signal
5) curve length of signal
6) area under signal
7) minimum subtracted from maximum

Low frequency cardiovascular

1) mean of signal
2) standard deviation of signal
3) minimum value of signal
4) maximum value of signal
5) curve length of signal
6) area under signal
7) minimum subtracted from maximum

Derivative of low frequency

1) mean of signal
2) standard deviation ofsignal
3) minimum value of signal
4) maximum alue of signal
5) curve length of signal
6) area under signal
7) minimum subtracted from maximum

Upper and lower respiratory

1) mean of signal
2) standard deviation of signal
3) minimum value of signal
4) maximum value of signal
5) curve length of signal
6) area under signal
7) average amplitude of peaks
8) minimum amplitude of peaks
9) derivative of the amplitudes of the peaks in the signal
10) number of peaks in the signal
11) inhalation/exhilation ratio
12) ratio of inhalation ratios before and after a question is asked
13) minimum subtracted from maximum

Table 5 List of time domain features

5.3 Feature Extraction Methods

To extract the following features which are listed in table 5, (respiratory 7, 8,9,10 ,11 and high frequeny cardiovascular 7, 8, 9), it was necessary to locate the peaks of the respiratory and the high frequency cardiovascular signals. This was not a trivial task because these signals contained low amplitude high frequency noise which was difficult to eliminate without distorting the data (see figures 8,10, and 12). In order to find the useful peaks, two programs were written. The program that found the peaks of the respiratory signal was titled peaklr and the program that found the peaks in the cardiovascular signal titled peakcard. Both programs can be found in the appendix. The way that these programs find peaks is as follows: The second derivative was taken and points that had values equal to zero were labeled as peaks. The amplitudes of the signal at points near these peaks were evaluated and the maximum of these values were labeled as peaks.

In order to eliminate the effects of the low amplitude high frequency noise, it was necessary to check the amplitude of data points that were near each point that had been labeled as a peak. The number of the data points from the peaks that were determined by the second derivative was chosen by examining many respiratory and cardiovascular signals and determining the average width of the peaks in these signals. It was found that twenty points on each side of the each peak found by the second derivative was a satisfactory range for the respiratory signals. Similarly eight points on each side of the initial peak gave would satisfy this criterion for the cardiovascular signal. All of the routines used to perform these operations are in appendix B (see peak.m, peakcard.m, and peaklr.m).

In order to determine the I/E ratio, it was necessary to find the valleys of the respiratory signals as well as the peaks. The method used to find the valleys was the same as that used to find the peaks (see appendix B valley.m and valleylr.m). The I/E ratio was found by the following method. First the time that a valley occurred was subtracted from the time that a peak occurred. Then the time that the peak occurred was subtracted from the time that the next valley occurred. The first value was then divided by the second value (see appendix B ie.m and ieie.m).

6.1 Conclusion

A vector of features was created by the program featurev.m which first executed all of the preprocessing routines. The program then extracted features for all of the questions using the times specified in table 4. This program extracted features from all polygraph files in a directory and produced a set of vectors. These vectors were then used for training and testing of a fuzzy K nearest neighbor classifier. For details on the methods used for training and testing as well as the frequency and correlation domain features used in the study refer to Dastmalchi [14]. For details on the K nearest neighbor algorithm refer to Layeghi [15].

REFERENCES

[1] Dale E. Olsen, et. al., "Recent developments in polygraph testing: A research review and evaluation - A technical memorandum, " Washington,DC: US Government Printing Office 1983.

[2] John C. Kircher and David C. Raskin, "Human versus computerized evaluations of polygraph data in a laboratory setting, " Journal of Applied Psycology, Vol.73, 1988 No2, pp291-308

[3] John E. Reid and Fred E. Inbau, Truth and Deception: The Polygraph (" Lie Dector ") Technique, The Williams & Wilkins Company, Baltimore, Md., 1966

[4] Michael H. Capps and Norman Ansley, "Numerical Scoring of Polygraph Charts: What Examiners Really Do", Polygraph, 1992, 21, 264-320

[5] L. A. Zadeh, "Fuzzy sets", Information and Control, vol. 8, pp.338-332, 1965

[6] James C. Bezdek and Sankar K. Pal, Fuzzy Models for Pattern Recognition Methods that Search for Structures in Data, IEEE Press, Piscataway, NJ. 1992

[7] L. A. Zadeh, "Calculus of fuzzy restrictions," in: L. A. Zadeh,K. S. Fu, K. Tanaka and M. Shimura, eds., Fuzzy Sets and Their Applications to Cognitive and Decision Processes, Academic Press, New York, 1975, pp 1-39

[8] Bart Kosko, Neural Networks and Fuzzy Systems, New Jersey : Prentice-Hall, Inc., 1992.

[9] Brian M. Duston, " Statistical Techniques for Classifying Polygraph Data ", Draft, November 24, 1992

[10] Howard W. Timm, " Analyzing Deception From Respiration Patterns " , Journal of Police Science and Administration, 1982, 1, 47 - 51.

[11] Personal communication with Richard Petty (polygraph examiner), June 1993

[12] Personal communication with Christopher B. Pounds (University of Washington), May 1993

[13] Personal communication with Howard Timm May 1993

[14] Mitra Dastmalchi , " Frequency Domain Features for Pattern Recognition of Polygraph Data", Masters Project, San Jose State University, November 15, 1993

[15] Shahab Layeghi, " Pattern Recognition of Polygraph Data", Masters Project , San Jose State University, November 15, 1993

Appendix A

Preprocessing Programs

```
function y = dercd(var)

% This extracts the derivative of a lowpass
% filtered version of the cardio signal.
%
% To use this command the user must enter the file name
%
% eg.   dercd(variable name)

q = detlc(var);  % detrends the lower frequencies
                 % of the cardio signal

e = diff(q);     % differentiates the lower
                 % frequencies of the cardio signal

x = e/std(e);

y = [x',x(length(x))']';
```

```
function y = detgsr(var)

% This function detrends the gsr
%
% To use this command the user must enter the file name
%
% eg.   detgsr(file name)

dtrnd = detrend(var(:,1));
                              % elliminates dc trends in signal
                              % eg. a line added to the signal
window = 100;

dtrnd = [dtrnd', zeros(window/2 - 1,1)']';
                    % adds zeros to end of signal so that no
                    % information is lost during filter delay


b = fir1(window,.03);
x = filter(b,1,dtrnd);
q = x/std(x);
l = length(q);

y = q(window/2:l);                % compensate for time delay
```

```
function y = dethic(var)

% This function detrendeds the high frequencies
% of the cardio signal.
%
% To use this command the user must enter the file name
%
% eg.   dethic(file name)

dtrnd = detrend(var(:,2));  % elliminates dc trends in signal
                            % eg. a line added to the signal

window = 134;

dtrnd = [dtrnd', zeros(window/2 - 1,1)']';
                        % adds zeros to end of signal so that no
                        % information is lost during filter delay


b = fir1(window,.035,'high');
                        % filter to elliminate low frequencies
x = filter(b,1,dtrnd);
q = x/std(x);

l = length(q);

y = q(window/2:l);          % compensate for time delay
```

```
function y = detlc(var)

% This function extracts and detrends the low
% frequencies of the cardio signal
%
% To use this command the user must enter the file name
%
% eg.   detlc(file name)

dtrnd = detrend(var(:,2));  % elliminates dc trends in signal
                            % eg. a line added to the signal
window = 134;

dtrnd = [dtrnd', zeros(window/2 - 1,1)']';
                          % adds zeros to end of signal so that no
                          % information is lost during filter delay

b = fir1(window,.035);    % filter to elliminate high frequencies
x = filter(b,1,dtrnd);
q = x/std(x);

l = length(q);

y = q(window/2:l);        % compensate for time delay
```

```
function y = detlr(var)

% This function extracts and detrends the lower respiratory signal
%
% To use this command the user must enter the file name
%
% eg.   detltr(file name)

dtrnd = detrend(var(:,4)); % elliminates dc trends in signal
                           % eg. a line added to the signal
window = 240;

dtrnd = [dtrnd', zeros(window/2 - 1,1)']';
                        % adds zeros to end of signal so that no
                        % information is lost during filter delay


b = fir1(window,.083);    % filter to elliminate noise
x = filter(b,1,dtrnd);
q = x/std(x);

l = length(q);

y = q(window/2:l);         % compensate for time delay
```

```
function y = detur(var)

% This function detrends the upper respiratory signal
%
% To use this command the user must enter the file
%
% eg.   detur(file name)

dtrnd = detrend(var(:,3));  % elliminates dc trends in signal
                            % eg. a line added to the signal
window = 240;

dtrnd = [dtrnd', zeros(window/2 - 1,1)']';
                          % adds zeros to end of signal so that no
                          % information is lost during filter delay


b = fir1(window,.08);              % filter to elliminate noise
x = filter(b,1,dtrnd);
q = x/std(x);

l = length(q);

y = q(window/2:l);                  % compensate for time delay
```

Appendix B

Feature Extraction Programs

```
function [x,y,z] = featurev(file_name,relevant,irrelevant,control,features)

% This function produces a feature vector for a given file
% Relevent, irrelevent, and control are vectors which contain
% the questions these features are extracted from.
%
% eg. featurev(t79,[3 5],[1 4], [6 10],feature_list)

% The above example gives the features for
% the file t79 of the 3rd and 5th question which are relevent in this
% MGQT format, the 1st and 4th question which are irrelevent
% and the 6th and 10th questions which are control

% feature_list=['10mean(frag ) ';
%               '20curve(frag )';
%               '30area(frag ) '];


feature_list = features



% The channels are ordered as follows:
% 1:GSR, 2:HiCardio, 3:LowCardio, 4:DerLowCardio, 5:LowResp, 6:UpResp


% This is a matrix of the time delay after asking a question to start of extracting
% the feature, and finish extracting the feature for each channel.

Times=[ 2, 14;
        3, 9 ;
        2, 18;
        0, 8 ;
        2, 18;
        2, 18];

% These are preprocessing functions.
Preprocess=[    'detgsr';
                'dethic';
                'detlc ';
                'dercd ';
                'detlr ';
                'detur '];

data=zeros(6,length(file_name(:,5)));
% Standardize and detrend the channels and derive new channels

for i=1:6,
        data(i,:)=eval([Preprocess(i,:),'(file_name)'])';
end
```

```
marker = file_name(:,5);    % 0 begin test and end test
                            % 0 examiner begins asking question
                            % 1 examiner finishes asking question
                            % 2 subject begins response to question
                            % 9 does not mark an event

begin = find(marker == 0);     % finds indecies where marker = 0 (question begins)
begin=begin(2:length(begin));      % elliminates the marker at the beginning of the test

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This for loop creates feature vectors for each relevant quesion
%
% eg x = [mean(gsr),std(gsr),area(gsr),mean(lr),std(lr),area(lr),etc.....
%           curve length,amplitude of peaks,# of peaks]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

feature_count=1;

for i = 1:length(relevant),
        question=relevant(i);

        for j=1:length(feature_list(:,1))
                channel_number=eval(feature_list(j,1));
                second_channel=eval(feature_list(j,2));
                st=begin(question)+30*Times(channel_number,1);
                fn=begin(question)+30*Times(channel_number,2);
          st2=begin(question)-30*Times(channel_number,2);
          fn2=begin(question)-30*Times(channel_number,1);
                fr=feature_list(j,3:length(feature_list(1,:)));
                frag=data(channel_number,st:fn);
                frag2 = data(channel_number,st2:fn2);
                if second_channel ~= 0
                        st3=begin(question)+30*Times(second_channel,1);
                fn3=begin(question)+30*Times(second_channel,2);
                        frag3 = data(second_channel,st3:fn3);
                end
                tempy=eval(fr);
          for m = 1:length(tempy)
             x(feature_count) = tempy(m);
                  feature_count=feature_count+1;
          end
        end
end
%-----------------------------------------------------------
% Irrelevant questions

feature_count=1;

for i = 1:length(irrelevant),
        question=irrelevant(i);

        for j=1:length(feature_list(:,1))
                channel_number=eval(feature_list(j,1));
```

```
                second_channel=eval(feature_list(j,2));
                st=begin(question)+30*Times(channel_number,1);
                fn=begin(question)+30*Times(channel_number,2);
            st2=begin(question)-30*Times(channel_number,2);
            fn2=begin(question)-30*Times(channel_number,1);
                fr=feature_list(j,3:length(feature_list(1,:)));
                frag=data(channel_number,st:fn);
                frag2 = data(channel_number,st2:fn2);
                if second_channel ~= 0
                        st3=begin(question)+30*Times(second_channel,1);
                fn3=begin(question)+30*Times(second_channel,2);
                        frag3 = data(second_channel,st3:fn3);
                end
                tempy=eval(fr);
            for m = 1:length(tempy)
               y(feature_count) = tempy(m);
                    feature_count=feature_count+1;
            end
           end
end

%-------------------------------------------------------
% Control questions

feature_count=1;

for i = 1:length(control),
        question=control(i);

        for j=1:length(feature_list(:,1))
                channel_number=eval(feature_list(j,1));
                second_channel=eval(feature_list(j,2));
                st=begin(question)+30*Times(channel_number,1);
                fn=begin(question)+30*Times(channel_number,2);
            st2=begin(question)-30*Times(channel_number,2);
            fn2=begin(question)-30*Times(channel_number,1);
                fr=feature_list(j,3:length(feature_list(1,:)));
                frag=data(channel_number,st:fn);
                frag2 = data(channel_number,st2:fn2);
                if second_channel ~= 0
                        st3=begin(question)+30*Times(second_channel,1);
                fn3=begin(question)+30*Times(second_channel,2);
                        frag3 = data(second_channel,st3:fn3);
                end
                tempy=eval(fr);
            for m = 1:length(tempy)
               z(feature_count) = tempy(m);
                    feature_count=feature_count+1;
            end
           end
end
```

```
function y = ampcard(var)

% This function finds the average of the amplitudes
% of the peaks in the high
% cardio signal over a specified period of time.
%
% To use this command the user must enter the
% file name and the start and finish points
% of the signal to be displayed
%
% eg.   ampcard(variable name)

p = peakcard(var);      % the indecies of the peaks

for n = 1:length(p)

   q(n) = var(p(n));    % amplitude of the peaks

end

y = sum(q)/length(q);
```

```
function y = ampr(var)

% This function finds the average of the
% amplitudes of the peaks in the lower
% respiratory signal over a specified period of time.
%
% To use this command the user must
% enter the variable name
%
% eg.   ampr(variable name)

p = peaklr(var);        % the indecies of the peaks

for n = 1:length(p)

   q(n) = var(p(n));    % amplitude of the peaks

end

y = sum(q)/length(q);
```

```
function y = curve(var)

% This function finds the length of the variable
%
% To use this command the user must enter the
% variable name and the start and finish points
% of the signal to be displayed
%
% eg.   curve(variable name)


x = sqrt(diff(var).^2 + 1);
y = sum(x);
```

```
function y = ie(var)

% This function takes the i/e ratio of the respiratory signals.
%
% To use this command the user must enter the variable name
%
% eg.   ie(variable name)

p = peaklr(var);        % finds the indices of
                        % the peaks in a signal and puts them
                        % in a vector a
plength = length(p);

v = valleylr(var);      % finds the indices of the
                        % valleys in a signal and puts them
                        % in a vector b
vlength = length(v);

if vlength < 2 | plength < 2        % check that enough peaks
                                    % and valleys exist for
                                    % the calculation to be done

   message = ' Warning !!!! Not enough data'

end

if p(1) > v(1)

   for n = 1:vlength - 1

       q = p(n) - v(n);             % calculates a vector of
                                    % e/i ratios for the given
                                    % time period
       z = v(n + 1) - p(n);

       e(n) = q ./ z;
   end

end

if p(1) < v(1)

   for n = 1:vlength - 1

       q = p(n + 1) - v(n);         % calculates a vector of
                                    % e/i ratios for the peaks
                                    % and valleys in the
                                    % given time period

       z = v(n + 1) - p(n + 1);
```

```
        e(n) = q ./ z;
    end
end
y = mean(e);
```

```
function y = ieie(var1,var2)

% This function takes the i/e ratio of the respiratory signals
% before and after a question is asked.  It then divides the two
% values.
%
% To use this command the user must enter the variable name
%
% eg.   ieie(variable name1, variable name2)

a = ie(var1);

b = ie(var2);

y = a/b;
```

```
function y = peak(var)

% This function finds the peaks in a signal and returns the index
% It also creates a plot of the variable with the peaks marked
%
% To use this command the user must enter the variable name
% of the signal to be displayed
%
% eg.   peak(variable name)

q = diff(var);      % differentiates the variable

z = q>0;            % z = 1 if q is greater than 0

f = diff(z);        % 2nd derivative of the variable

a = f<0;

y = find(a);        % finds the indices where the 2nd derivative
                    % is -1 which indicates peak
```

```
function y = peakcard(var)

% This function finds the peaks in
% the cardio signal and returns a vector of
% indexes where they occur.
%
% To use this command the user must enter the variable name
%
% eg.   peakcard(variable name)


ty = peak(var);

if ty(1) < 8
        ty = ty(2:length(ty));
end

if ty(length(ty)) > length(var) - 8
        ty = ty(1:length(ty)-1);
end

for n = 1:length(ty);
                          % finds the maximum peak over a 10 point s
pan


     temp = var(ty(n)-8 : ty(n)+8);

     z(n) = ty(n) - 9 + find(temp == max(temp));
                                % finds the time that the peak
                                % occurs in the original signal
end

for n = 1:length(z)-1  % elliminates duplicate indicies

    if z(n) == z(n+1)

          z(n) = 0;

     end

end

ind = find(z);                % finds indecies of elements
                              % that are not equal to zero

for n = 1:length(ind)         % elliminates 0 elements

     z(n) = z(ind(n));

end
```

```
function y = peaklr(var)

% This function finds the peaks
% in the lr signal and returns a vector
% of indecies where they occur.
%
% To use this command the user must enter the variable name
%
% eg.   peaklr(variable name)


[b,a] = butter(4,.034);              % elliminate noise
filtout = filtfilt(b,a,var);

ty = peak(filtout);  % finds the time that the
                     % peaks of filtered lr signal occur

if ty(1) < 20
        ty = ty(2:length(ty));
end

if ty(length(ty)) > length(var) - 20
        ty = ty(1:length(ty)-1);
end

for n = 1:length(ty)

     temp = var(ty(n)-20:ty(n)+20);
     z(n) = ty(n) - 21 + find(temp == max(temp));
                       % finds the time that the peak occurs in
                       % the original signal
end

for n = 1:length(z)-1          % elliminates duplicate indicies

    if z(n) == z(n+1)

          z(n) = 0;

     end

end

ind = find(z);          % finds indecies of elements
                        % that are not equal to zero

for n = 1:length(ind)  % elliminates 0 elements

     z(n) = z(ind(n));

end
```

```
y = z(1:length(ind));

% pmark = zeros(1,length(var)); % a vector of 1's where peaks occu
r
                                % 0's everywhere else
% pmark(y) = ones(1,length(y));

% plot(var,'r')

% title('lr marked with peaks')

% hold on

% plot(5*pmark,'g')

% hold off
```

```
y = z(1:length(ind));
```

```
function y = peaknumc(var)

% This function finds the number of
% peaks in the high cardio signal
%
% To use this command the user
% must enter the variable name
%
% eg.   peaknumc(variable name)

p = peakcard(var);          % the indecies of the peaks

y = length(p);
```

```
function y = peaknumr(var)

% This function finds the number
% of peaks in the respiratory signal
%
% To use this command the user
% must enter the variable name
%
% eg.   peaknumr(variable name)

p = peaklr(var);          % the indecies of the peaks

y = length(p);
```

```
feature_list=[  '10mean(frag)              ';
                '10curve(frag)             ';
                '10area(frag)              ';
                '20mean(frag)              ';
                '20curve(frag)             ';
                '20area(frag)              ';
                '20ampcard(frag)           ';
                '20peaknumc(frag)          ';
                '30mean(frag)              ';
                '30curve(frag)             ';
                '30area(frag)              ';
                '40mean(frag)              ';
                '40curve(frag)             ';
                '40area(frag)              ';
                '50mean(frag)              ';
                '50curve(frag)             ';
                '50area(frag)              ';
                '50ampr(frag)              ';
                '50peaknumr(frag)          ';
                '50ie(frag)                ';
                '50ieie(frag, frag2)       ';
                '60mean(frag)              ';
                '60curve(frag)             ';
                '60area(frag)              ';
                '60ampr(frag)              ';
                '60peaknumr(frag)          ';
                '60ie(frag)                ';
                '60ieie(frag, frag2)       '];
[x y z] = featurev(t79,[1 2],[3 4],[6 10],feature_list)
```

```
function y = valcard(var,start,finish)

% This function finds the valleys in
% the lr signal and returns a vector of indexes where
% they occur
%
% To use this command the user must enter the
%  file name and the start and finish points
% of the signal to be displayed
%
% eg.   valcard(file name, start, finish)


k = hicardio(var,start,finish);



[b,a] = butter(4,.034);         % elliminate high frequencies
filtout = k; % filtfilt(b,a,k);

ty = valley(filtout,start,finish) % finds the time that the
                                  % peaks of filtered lr signal oc
cur

l = length(ty);


for n = 1:l

     temp = k(max(1,ty(n)-10+start) : min(ty(n)+10+start,length(k)
));

     if ty(n)<10
        dd=length(temp)/2+1;
     else
        dd=11;
     end

     y(n) = ty(n) - dd + find(temp == min(temp));
                         % finds the time that the peak occurs in
                         % the original signal
end

vmark = zeros(1,finish - start); % a vector of 1's where peaks occ
ur
                                 % 0's everywhere else
vmark(y) = ones(1,length(y));

subplot(211),plot(k(start:finish),'r')
```

```
title('lr marked with peaks')
hold on
plot(-5*vmark,'g')
hold off
subplot(212),plot(filtout(start:finish),'r')
title('filtered lr marked with peaks')
hold on
plot(vmark,'g')
hold off
% subplot(223),plot(k(start:finish),'r')
% hold on
% plot(5*a(1:finish - start - 3),'g')
% hold off
% subplot(224),plot(x)
% subplot(111)
```

```
function y = valley(var)

% This function finds the
% valleys in a signal and returns the index

% To use this command the user
% must enter the variable name
%
% eg.   valley(variable name)

q = diff(var);          % differentiates the variable

z = q > 0;      % z = 1 if q is greater than 0

f = diff(z);    % 2nd derivative of variable

a = f > 0;      % finds valleys

y = find(a);    % finds the indices where the 2nd derivative
                % is +1 which indicates valleys
```

```
function y = valleylr(var)

% This function finds the valleys in
% the lr signal and returns a vector of
% indecies where they occur
%
% To use this command the user must enter the variable name
%
% eg.   valleylr(variable name)


[b,a] = butter(4,.034);       % elliminate high frequencies
filtout = filtfilt(b,a,var);

ty = valley(filtout);    % finds the time that the
                         % peaks of filtered lr signal occur

for n = 1:length(ty)

     temp = var(max(1,ty(n)-20) : min(ty(n)+20,length(var)));

     if ty(n)<20
        dd=length(temp)/2+1;
     else
        dd=21;
     end

     z(n) = ty(n) - dd + find(temp == min(temp));
                         % finds the time that the peak occurs in
                         % the original signal
end

for n = 1:length(z)-1    % elliminates duplicate indicies

    if z(n) == z(n+1)

          z(n) = 0;

     end

end

ind = find(z);           % finds indecies of elements
                         % that are not equal to zero

for n = 1:length(ind)    % elliminates 0 elements

     z(n) = z(ind(n));

end
```

```
y = z(1:length(ind));
```

A Comparison of Fuzzy Logic Algorithms for Pattern Recognition

Shahab Layeghi
Electrical Engineering Department
San Jose State University
Professor: Ben Knapp

December 1993

I. Introduction

A great amount of work has been done on the application of fuzzy logic techniques for pattern recognition. In this study some of the more important algorithms are summarized and compared.

Pattern recognition could be defined as search for structure in data. This means organizing data in groups in a way that members of each group have some kind of similarity. A system that does this job is called a classifier. A classifier can be designed by a human expert and be used to classify the data (fixed design). Another approach is to provide the classifier with the data and make it adapt itself according to the data that it receives. Adaptive systems can be divided into two main categories, supervised and unsupervised.

In supervised learning, another system (or a human expert) which is usually called a teacher, furnishes the classifier with the group that each data item belongs to, so that classifier can learn from a set of labeled input data and be able to classify new data. This process is called training.

In unsupervised learning, which is also called clustering, the system is given a set of unlabled data, and it is expected that it find internal similarities between the data items and put them in different groups accordingly. If data are represented quantitatively as vectors in a vector space, data that are spatially close should be put in one group.

In the section ,a method of classification is described which uses fuzzy linguistic variables. This method uses human experts to train the system and then uses the labeled linguistic samples to refine the classifier. In section 2, C-Means Algorithm which is a clustering method is explained. Section 3 covers K Nearest Neighbor algorithm which is a supervised classification method.

Polygraph Classification Project: A Brief Guide

This is an informal report about the Polygraph Classifier project at San Jose State university during May to December 1993. The purpose of writting this report is to help the persons that follow the project to have a quick understanding of the practical issues in the project. It is assumed that they have studied or have access to the reports of Eric, Mitra, and Shahab, and other papers about the polygraph, and they refer to the programs source codes whenever necessary.

Reading the data:

Polygraph tests are supplied in the format of dos files by the NSA. Each polygraph test may consist of one to five charts. Each chart is a series of questions, usually ten questions. For each chart three files exist. Before being able to see the files, the files should be read and decoded. The files are given as compressed and backed up dos files. In order to restore the files, the first floppy diskette for each series, should be put in the drive and be restored by using a dos command like:
restore b: c: /S
This will copy the file into a pre specified directory in hard drive. The next step is to decompress the files. The files are compressed using PKZIP version 1.01. The PKZIP and PKUNZIP programs are supplied with the data files. With each series of files, appropriate instructions on how to uncompress them, and a listing of the files are given. Also for most of the files, there is a classification sheets by the corresponding agency which shows the scoring by the polygraph examiner.
The files that comprise a chart look like this:

$$7%du1x.011
$$7%du1x.012
$$7%du1x.013

Each of the above mentioned files have a specific significance. The .xx3 files are text files that contain questions. The .xx1 and .xx2 files are encoded in a special format created by axciton polygraph machines. These files can be decoded by a program called read3. read3.exe is the executable code of a C program written by other groups and modified by Shahab. read3 can be invoked as in the following example:

read3 $$7%du1x.011 output

The line above decodes the information in files x.011 and x.012 and writes it as an ASCII file called output. This file contains the actual signals from four polygraph channels and a timing signal that shows the times that a question is asked. For more information about the format of this file refer to the correspondence from Chris Pounds which are in the

directory \polygrap\project\source and saved as Postscript files: chris.ps and mail.fil . One of these files is the first year report of Chris Pounds group to NSA, and the other one is an explanation of the above mentioned file. Printed versions of these and also reports from other groups can be found in the lab. There is another file in the same directory called fred.txt which is the correspondence between Chris and Dr. Knapp about obtaining sample files and decoding them which is not very important.

The data signals can be plotted by writing a simple Matlab routine. The routine should read the ASCII file and plot the columns 1 to 4. The fifth column is the timing marker which can be used to mark the start of questions. Another way to see the data is to use the APL program.

A program is written by Johns Hopkins APL group which reads the Axciton files and classifies them. This is the commercial program that is currently used for automatic scoring of polygraphs. The program is able to read the data, plot them, print them and finally Classify the subjects. This program can be used to test the data files before using them in the project and classify them for comparison purposes. For more information about the program refer to the user manual. The program can be found in the directory \polygrp\axciton, and also in the directory \users\nsa\axciton in the hp 486. The polygraph files are stored in the subdirectories of the directory \mgqt in the hp 486. They are organized according to the test format and polygraph examiner.

As mentioned above, the polygraph data were restored and unzipped and put on the hard disk of the computer. The read3 program decodes a single file. In order to decode all the files in a directory a C routine was written that invoked the read3 program for each file in a directory and saved the result as a file with the same name but starting with qq instead of $$. This program is called decode. The following is repeated from the source code of the decode program:

/* This program decodes all the axciton files in a directory by running the decoder program 'read3' for each file. The result of decoding the files:
$$name.xx1, $$name.xx2 , $$name.xx3
is an ASCII file named qqname.
This program searches in the current directory for the files that start with '$$'. The pathname can also be given in the command line.

Ex.:

```
decode c:\axciton\$$*.*
```

The decoded files go to the current directory. The read3 program should be in the directory of the files otherwise it doesn't work.
*/

Feature Extraction:

After polygraph files were decoded and put in a directory, they could be processed using Matlab. It was tried to write the programs in a structured way so that creating and debugging of individual sections would be easier and program segments would have direct conformity with conceptual block diagrams. At the lowest levels, there are many Matlab routines that operate on pieces of data and extract features from them, and return these features to the calling routines. At the top, there is a Matlab program that extracts the features for all the files in a directory and saves it as a matrix. The structure of these programs is explained in the following sections.

The main feature extraction program is a Matlab routine called **newfeat**. This program finds the features for the files in a directory and saves the features in a matrix. The main part of the program is a loop that extracts the features of a single file and puts them in a vector. This action is repeated for all the files that their name is given. In order for the Matlab program to find the files to processed in a directory, a C program was written that searches in a directory and saves all the names of all the files that it finds in an ASCII file containing a Matlab matrix. This C program is called '**flist**' and could be found in the \polygrap\project\source directory. The way this program works is explained below:

/* This program lists the files in a dos directory and saves this
listing in a file called **files.m**. This file is actually a Matlab script
that contains a matrix called '**flist**' which holds a filename in each row.
The first character of file names can be given to this program as an
input argument.

Ex:
flist t
is equal to use the dos command
dir t*.*
and save the result in a Matlab m file called files.m

*/

After running the flist.exe program in a directory, and checking that the appropriate filenames are saved in the files.m file, the Matlab program can use them by executing the command
files
and using the variable flist.

Another important data item that is used in the feature extraction programs is called **feature_list.** It is a Matlab matrix that includes the names of feature extraction routines. In each row of the feature_list matrix a feature extraction routine is named along with the channel number(s) that this routine will be applied to. For example

'10mean(frag)'

means to apply the mean function to a piece of data called frag, which is defined later. The channel that data is to be gathered from is channel 1. As another example

'26crosscor(frag, frag3)'

means to apply the function crosscor to two pieces of data coming from channels 2 and 6, in variables called frag and frag3.

feature_list is defined in **newfeat** program. All the features that are extracted from the data are listed in it. If a new feature is to be investigated, it is enough to write a program that extracts it, and add that program name in this list.
Note: It is highly recommended that the programs newfeat, feature, and processf be read carefully before making any changes in feature list.

Before being able to do any processing on the data, for each data file another file should be created that holds the types of the questions. These files are named ***zzname*.0x4**. Note that these files are not a standard part of axciton files and were created here by referring to the question files and data sheets that accompanied each the files. The format of these files is as follows:

```
x  0  0  0  0
a1 b1 c1 d1 e1
a2 b2 c2 d2 e2
a3 b3 c3 d3 e3
```

x is either one or zero. 1 means the file is deceptive, and zero means it is non-deceptive. The rows 2, 3 and 4 in this file show the numbers of relevant, irrelevant, and control questions. For example for a deceptive file in which questions 3, 5, 8 and 9 are relevant, questions 1, 2, 4, and 7 are irrelevant, and questions 6 and 10 are control, a question file is constructed that looks like this:

```
1  0  0  0  0
3  5  8  9  0
1  2  4  7  0
6 10  0  0  0
```

The **newfeat** program, for each data file which is listed in flist, loads the above mentioned question file to find the question types. Then it calls the actual feature extraction routine which is called **feature.** The program feature finds all the features for each relevant, irrelevant, and control question and returns the results in a vector. This vector is added as a new column to a matrix called M. At the end of the newfeat program the matrix M is saved in a file. This file is manipulated by another program called **processf**.

processf is a program that loads the M matrix, combines the features for each question in different ways that are explained in reports of Mitra and Shahab, and saves the resultant matirx, the F matrix, in a file.
The above procedure was repeated for the polygraph files in several directories. One of the directories contained files for non-deceptive cases and the other ones included deceptive files. Three sets of data were built by combining the features for non-deceptive cases with three sets of deceptive files. Each data set contained 50 deceptive and 50 non-deceptive cases. These sets were used by classification programs.

Classification:

There are two classifier programs written for this project, fknn and cknn, which implement fuzzy and crisp K-nearest neighbor classifiers accordingly. These programs are written in C programming language. The way they interact with Matlab is through reading and writing files in Matlab format, that is **.mat** files. There are two C functions inside these programs called loadmat and savemat which are interfaces to Matlab files and can be used to load and save date, which in Matlab are matrices, from Matlab files. These two functions are in a file called **matldsv.c** which should be compiled with the source files that use them. fknn and cknn programs load matrices that include the features and were prepared by Matlab feature extraction routines. After loading the matrices, the feature vectors in test matrix are classified individually, and the result is saved in a file as a Matlab matrix. The comments in the source codes of the programs cknn and fknn are repeated here for reference:

/* **cknn**: This program implements a K-nearest neighbor classifier.

The main program opens a Matlab data file, reads the training matrix,
classifies each entry in the testing matrix, and writes the result in an
output file. The file that this program gets the information from should be
called "cdatafil.mat". As the name implies it is in Matlab file format.
The data in this file should have the following order:

1. A single variable 'C' which is the number of classes.
2. A single variable 'K' which is the parameter 'K' in K-NN Algorithm.
3. A training matrix 'P' which contains a set of feature vectors. Each vector
is in a column of the matrix.
4. A classes vector 'T' which contains the classes of the training set
5. An input matrix 'U' which contains a set of unclassified feature vectors.

The main program uses the Crisp KNN routine to classify each one of the input
vectors and saves the results (the classes that these inputs belong to) in a
file called coutfile.mat. This file is in Matlab format. This file contains
a vector of the classes called:

'cresult'

This program can be called from dos, or within Matlab by using dos escape character '!'. An example Matlab script file that shows how this program can be used is included in the file "cknntest.m".

*/

/* **fknn**: This program implements a fuzzy version of K-nearest neighbor classifier.

The main program opens a Matlab data file, reads the training matrix, classifies each entry in the testing matrix, and writes the result in an output file. The file that this program gets the information from should be called "fdatafile.mat". As the name implies it is in Matlab file format. The data in this file should have the following order:

1. A single variable 'C' which is the number of classes.
2. A single variable 'K' which is the parameter 'K' in K-NN Algorithm.
3. A single variable 'M' which is the coefficient in fuzzy algorithm.
4. A training matrix 'P' which contains a set of feature vectors. Each vector is in a column of the matrix.
5. A class membership matrix 'T' which contains the membership values of the training set vectors to the classes.
6. An input matrix 'U' which contains a set of unclassified feature vectors.

The main program uses the Fuzzy KNN routine to classify each one of the input vectors and saves the results (the memberships of the inputs to classes) in a file called "foutfile.mat". This file is in Matlab format. This file contains a single variable called fresult. It is a matrix of the memberships of the inputs to the classes.

This program can be called from dos, or within Matlab by using dos escape character '!'. An example Matlab script file that shows how this program can be used is included in the file "fknntest.m".

*/

As mentioned above, the programs fknn and cknn are the actual classifiers which can be called directly from dos or within a Matlab program. Several Matlab programs were written that used these two programs for classification of data. The Matlab programs acted mostly as a front end or user interface for the classifier programs. A listing of many Matlab programs and functions is included as an appendix in this report. Understanding of all the functions is not necessary because they are used inside the programs. Some of the

programs were created to test other programs or to experiment with the data. These programs are not necessary for classification, but knowing about them might help to prevent recoding routines that are already there. In the case of user interface programs, the best way is to run them and become familiar with the way they work. They were intended to be very flexible, and usually by changing a few parameters inside the code, they can be used for other purposes.

Classifier programs were used not only to classify a given data set, but also to select a set of good features from all the features that initially were tried. For a detailed discussion of the steps involved in this refer to Shahab's and Mitra's reports. Some of the programs and data files which were used or produced in this stage are explained here:

Classify is a Matlab program that loads a feature matrix from a .mat file, randomly breaks it into a set of training, and a set of testing feature vectors, classifies every entry in the testing set using all the entries in the training set by calling either fknn or cknn programs, repeats this process a number of time, and returns the result of classification of each file and the percentage of correct classification and a performance index for the classification. Some of the parameters like the filename to load can be changed inside the program classify.m. Other parameters can be changed while the program is running. This program is extremely useful for experimenting with combinations of features, and even includes an option to plot the scattering of the first two features.
Note: By setting percent_training=1, The testing and training sets wont be randomly selected, instead, all the entries except one are used in the training set and that entry is classified. This action is repeated for all the entries in the matrix.

Clas_aut is an automated version of classify program. Instead of asking the user for entering parameters, this program includes a loop that checks the classifications using all the features individually. The results are saved in a file called **clas_res**. All the other parameters should be set in the program. It should be noted that running this program might take a long time depending on the number of features and repetitions. **Clasaut2, clasaut3**, and **clasaut4** are alterations of clas_aut that instead of using single features use combinations of 2 to 4 features. Clasaut2 tries all the pairwise combinations of the features. Clasaut3 and clasaut4 use the combinations of 2 and 3 features supplied to them in the program and combine them with other features to test the combinations of 3 and 4 features.

bestfk is a Matlab script that sorts the features according to their performance in classifying the files. Note that the correct_classification vectors for data sets 1-3 were saved as res1, res2, and res3 in a file called **Knn-res**. This file is loaded by the bestfk program. The best features are found for the three data sets. For more details about the selection strategy refer to Shahab's report. It is informative to look at the program code to find out about the outputs that it produces.
Bestfs is the same program as bestfk. The only difference is that it loads the results from a file called **scat_res**. This file is produced by saving the results of the scatter criterion.

Scat is a Matlab function that finds the scatter criterion for the feature vectors in a matrix. It was used for the feature matrices of sets 1-3, and the results were saved in scat_res. **Bestf2**, **bestf3**, and **bestf4** work the same way as bestfk, but as output give the best combinations of 2-4 features.

Appendix: A listing of the Matlab programs

bestf2:
This Matlab script finds the best 30 combinations of features from three sets of features. Same features are tried on 3 sets of data
This is used to rank the combinations of 2 features

---

bestf3:
Same as bestf2, but for combinations of 3 features.

---

bestf4:
Same as bestf2, but for combinations of 4 features.

---

bestfs:
This Matlab script tries a method to find the best 30 features from three sets of features. Same features are tried on 3 sets of data. Scatter criterion is used to measure each feature's performance.
Note that for the features 1-651 each set of seven features are in fact the same feature combined differently for different features. For the rest of the features i.e. 652-669 each set of three is the same feature.

---

bestfk:
This Matlab script tries a method to find the best 30 features from three sets of features. Same features are tried on 3 sets of data. The results of classification using a KNN classifier is saved on a vector called correct_res.
Note that for the features 1-651 each set of seven features are in fact the same feature combined differently for different features. For the rest of the features i.e. 652-669 each set of three is the same feature.

---

clas_aut
This program adds a loop to the classify program. It repeats classification for different input vectors. It saves the results (percentage correctly classified and performance index)
as two vectors in a file called clas_res.

---

clasaut2:
This program adds a loop to the classify program. It repeats classification for different combinations of 2 features. It saves the results (percentage correctly classified and performance index) and the indexes of these features
in a file called clasres2.

---

clasaut3:
same as clasaut2, but for the combinations of 3 features.

---

clasaut4:
same as clasaut4, but for the combinations of 4 features.

---

classify:

This script parses a matrix of polygraph
vectors into training and testing vectors.
It then calls the classifier, trains, tests
and gives results.

---

cluster1:
This is a program that tests the K-Nearest-Neighbor
algorithm with a set of two class data that have
gaussian distribution

---

cluster2:
Another program like cluster1.

---

feattst:
An older version of classify.

---

feattst2:
Another version of feattst.

---

featurev:
Mitra

---

plotf:
This script prompts the use to enter two features and plots them.

---

randvect:
function [y,x] = randvect(elements,maximum)

This function creates a vector
of random numbers between 1
and the maximum number given
to the function (maximum).
The length of the vector is
specified by the number of
elements given to the function.
e.g. randvect(elements,maximum)

---

scat:
function J=scat(Sample, Class)

J=scat(Sample, Class)
returns a value that shows how the labeled samples of a two class distribution are scattered. Samples is a vector that contains the values of the samples. Class is a vector that contains the class labels(0 or 1).
The criterion function is:

J=(m1-m2)^2 / s1^2+s2^2
m's are the means for the classes and S's are scatters of samples.
Larger result means better separation between the classes.
Reference: Pattern Classification and Scene Analysis, Duda and Hart

---

scatv

function JV=scatv(M, Class)
scatv returns a vector that contains the scatter criterion of a matrix.
each row of the matrix M contains values of the samples for one feature.
Class is the class labels for the samples.
see also scat

Pattern Recognition of the Polygraph Using Fuzzy Set Theory

A Report
Presented to
The Faculty of the Department of Electrical Engineering
San Jose State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

By
Shahab Layeghi

December 1993

Contents:

	Page
I. Introduction	2
II. Polygraphs	4
III. Feature Extraction and Classification	7
IV. Conclusion and Future Work	28
References	29
Appendices	
A. Tables	
B. Program Listings	

I. Introduction

Polygraph examinations are the most widely used method to distinguish between truth and deception. In a Polygraph examination a person is connected to a special instrument called a Polygraph which records several physiological signals such as blood pressure, Galvanic Skin Response, and respiration. The subject is asked a set of questions by an examiner. By looking at these signals the examiner is able to determine the reactions of the subject to the questions and decide whether the person was truthful or deceptive in answering each question. The problem with human classification of Polygraph tests is that the outcome depends on the examiner's experience and personal opinion. Automatic scoring of Polygraph tests has been a subject of extensive research. Several methods for Polygraph classification have been studied which are mostly based on statistical classification techniques.

In this study two main goals were presented. The first goal was finding appropriate features which have physiological basis. The second purpose was trying a new classification method based on fuzzy set theory. The advantage of using fuzzy logic is that the it does not simply assigns each input to one of the classes, but it gives the possibility of belonging of an input to each class.

Digitized Polygraph data used in this project were collected from various police stations. The data files were organized according to the test format used and were decoded to ASCII format so they can be read by Matlab. Preprocessing and feature extraction routines were implemented in the Matlab language. Three sets of files were chosen, each one of them contained 50 deceptive and 50 non-deceptive files. These files are listed in Table 10 in Appendix A. A set of features were selected based on physiological reactions, and the feature vectors for every file in each set were found. Different classification methods were studied and a Fuzzy K-nearest neighbor classifier was selected. Significance of each feature was examined according to the clustering and correct classification obtained by using that individual feature. Thirty features were selected as the final set of features and a subset of combinations of 2 to 4 of these features were examined to study the effects of combining the features on classification results. The

combination that produced the best classification for all three sets on the average was selected and the effects of changing the classifier parameters on classification was studied.

II. Polygraphs*

A polygraph examination is the most popular method used to determine if an individual is being truthful or deceptive. During an examination, a subject is asked a series of questions and the physiological responses to the questions are recorded using a polygraph. The three physical responses currently obtained from a polygraph examinations are blood pressure, respiration, and skin conductivity. Polygraph charts are usually analyzed by a human interpreter for evidence of truth or deception; however, computer algorithms are now being used to verify results [1][2].

II.1. History

The first attempt to use a scientific instrument in an effort to detect deception occurred around 1895 [3]. That was the year that Caesar Lombroso published the results of his experiments in which a hydrosphygmograph was used to measure the blood pressure-pulse changes of criminals in order to determine whether or not they were deceptive. Although the hydrosphygmograph was originally intended to be used for medical purposes, Lombroso found that it worked well for lie detection. Lombroso may have been the first to use a peak of tension test format. This was done by showing a suspect a series of photographs of children, one being the victim of sexual assault. If the suspect did not react more to the victims picture than the pictures of the other children, Lombroso concluded that the suspect did not know what the victim looked like and therefore was not the alleged perpetrator.

In 1914 Vittorio Benussi published his research on predicting deception by measuring recorded respiration tracings [4]. He found that if the length of inspiration were divide by the length of expiration, the ratio would be larger after lying than before lying and also before telling the truth than after telling the truth. In 1921 John A. Larson constructed an instrument capable of simultaneously recording blood pressure pulse and respiration during an examination [3][4]. Larson reported accurate results which prompted Leonarde Keeler to construct a better version of this instrument in 1926 [3][4].

* This section is exerpted from [17]

The use of galvanic skin response in lie detection began during the turn of the century. It's usefulness, however, did not become evident until the 1930's during which time several articles written by Father Walter G. Summers of Fordham University in New York [4]. In these articles he reports over 90 criminal cases in which examination using the galvanic skin response had all been successful and confirmed by confession or supplementary evidence. The usefulness of the galvanic skin response prompted Keeler to add an galvanometer to his polygraph. At the time of Keelers death in 1949, the Keeler Polygraph recorded blood pressure-pulse, respiration, and galvanic skin response [3].

II.2 Modern Test Formats

The effectiveness of a polygraph examination is often the result of the test format that is used. A polygraph test format consists of an ordered combination of relevant questions about an issue, control questions that provide a physical response for comparison, and irrelevant questions that also provide a response or the lack of a response for comparison [1][4]. Three general types of test formats are in use today. These are Control Question Tests, Relevant-Irrelevant Tests, and Concealed Knowledge Tests. Each of the general test formats may have a number of more specific variations. Each test consists of two to five charts containing a prescribed series of questions. The test format that is used in an examination is determined by the test objective [3][4].

The concealed knowledge test, also called peak of tension test, is used when facts about a crime are known only by the investigators and not by the public. In this case, a subject would not know the facts unless he or she was guilty of the crime. For example, if a gun was used in a crime and the public did not know the caliber, an examiner could ask a suspect if it was a 22 caliber , a 38 caliber, or a 9 mm. If the gun used was a 9 mm and the suspect was deceptive, a polygraph chart would probably indicate evidence of deception.

A control question test is often used in criminal investigations. In this type of test a series of relevant, irrelevant, and control questions are asked. A relevant question is one which is specific to the crime being investigated. For example, " Did you steal the money?". A control question is designed to make the subject feel uncomfortable. It is not specific to the crime being investigated however it may be related in an indirect way. A control

question that could follow the relevant question stated above is "Have you ever taken anything that did not belong to you?". The control questions are compared to the relevant questions and if the responses to the relevant questions are greater, the subject is usually classified as deceptive. Irrelevant questions are used as buffers. Examples of irrelevant questions are "Are the lights in this room on?" or "Is today Monday?".

Relevant-Irrelevant tests are usually used to test people trying to obtain security clearance or get a job. In this test, relevant questions are compared to irrelevant questions. Very few control questions are asked. The purpose of control questions in this test is to make sure that the subject is capable of reacting at all.

II.3 Present Day Equipment

The most popular polygraph machines today are the Reid Polygraph developed in 1945 and the Axciton Systems computerized polygraph developed in 1989 [1][11]. The Reid polygraph scrolls a piece of paper under pens that record the biological signals. The Axciton polygraph digitizes physiological signals and uses a computer to process them. The sampling frequency of the Axciton machine is 30 Hz. Axciton provides a computer based system for ranking the subject responses but allows printouts of the charts to be scored by hand the traditional way. Both machines record the same biological signals using standard methods. Blood pressure is measured by placing a standard blood pressure cuff on the arm over the brachial artery. Respiration is monitored by placing rubber tubes around the abdominal area and the chest of the subject. This results in two signals, an upper and lower respiratory signal. Skin conductivity is measured by placing electrodes on two fingers of the same hand.

III. Feature Extraction and Classification

III.1 Introduction

The problem of Classification of Polygraph data like other pattern recognition problems can be considered of consisting of several main stages. Figure [1] shows these stages and the relationship between them. At the beginning data is preprocessed so that noise and redundancies are removed from data and feature extraction can be done more accurately. The next stage is feature extraction. In this step data is read and appropriate features are extracted from it. This is a very important step in all pattern recognition problems, because the purpose of pattern recognition is finding similarities in data that belong to the same class, and features are elements that represent these similarities. Therefore, a good set of features can lead to good classification whereas a satisfactory result cannot be achieved with an inappropriate set of features. Having a set of features, the next step is to use a method to classify data using these features. These steps as applied to Polygraph classification are described in more details in the following sections.

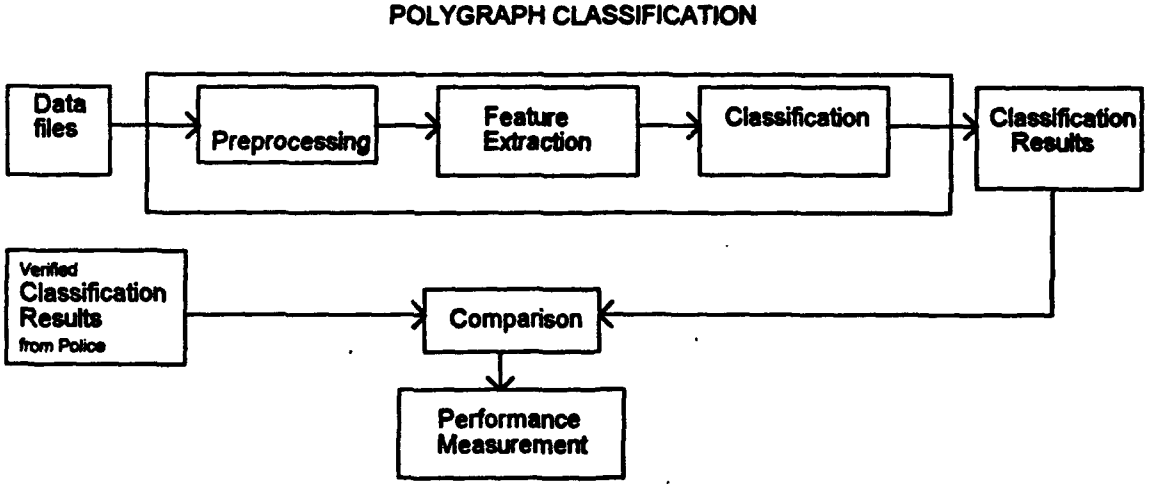


Figure 1

III.2. Preprocessing

Polygraph data consists of signals from four different channels: galvanic skin response (GSR), blood pressure, higher respiration, and lower respiration. First blood pressure signal was decomposed into a high frequency component showing heart pulse, and a low frequency component showing blood volume. Derivative of the blood volume channel was taken and used as another channel. These six derived signals were detrended and filtered. For more details on preprocessing refer to [17].

III.3. Feature Extraction

In this step appropriate features are selected and extracted. Feature extraction is itself divided into several steps. Figure [2] shows different stages involved in feature extraction.

By feature gathering we mean selecting features that might have useful information in them. Feature Combination is a special step in polygraph classification. In this step features derived for different questions in a test are combined to build a single feature. feature selection is a step in which a small number of features is selected from the main feature set to be used in final classifier section.

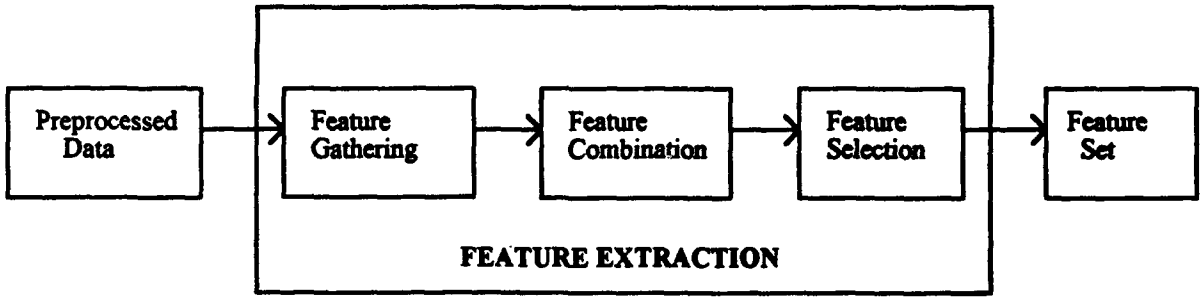


Figure 2

III.3.1. Feature Gathering

Features that possibly convey some information in them were selected and extracted in this stage. Literature about Polygraph were studied and several Polygraph examiners were interviewed to find out what had been done about this problem and what characteristics in a signal are used as indicators of truth or deception. In general features are divided into three main groups, time domain features, frequency domain features and correlation features. Time domain features are mostly standard characteristics like mean, standard deviation, median and so on. Some more specific time domain features were also added, such as the ratio between inhalation and exhalation. Auto Regressive parameters were also extracted and tried as features. To extract each feature for each question a time frame was considered that started with a specific delay after each question was asked and lasted for a specific amount of time. Different time frames were used for different channels because each channel represents a different physiological parameter. Frequency domain features include fundamental frequency, magnitude of power spectral density at fundamental frequency, coherency at fundamental frequency and so on. Figure 3 shows the feature gathering and the decisions that involved in this step.

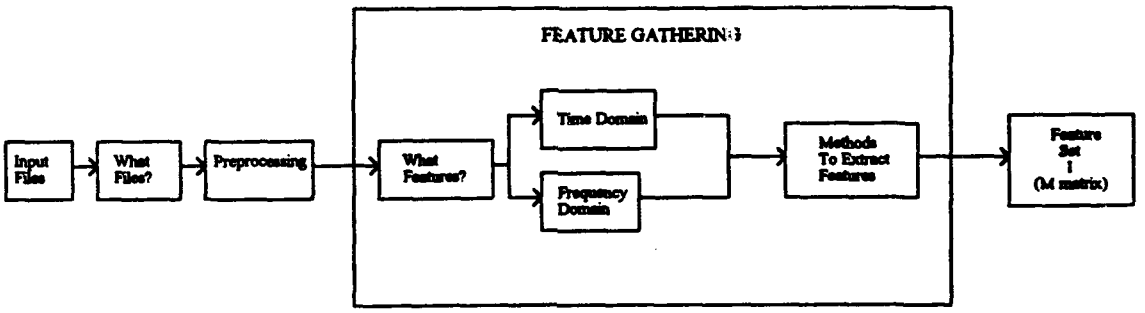


Figure. 3

For every question in a test 93 features were selected and extracted . Also 6 Integrated Spectral Density features were used which directly compare each relevant question to the nearest control question. The total number of features derived for each test was :

$$93 \times 10 + 6 \times 5 = 960$$

This was repeated for all the tests in feature sets 1, 2 and 3. The results of each set were saved in a 960x100 matrix called the M matrix.

For a detailed description of time domain features and frequency domain features refer respectively to [17] and [16].

III.3.2. Feature Combination

As mentioned earlier each feature is extracted for all questions in a test, that is for relevant, irrelevant, and control questions. In a polygraph test responses to relevant questions are compared to responses to irrelevant and control questions. But in any test there are several questions of each type and many methods can be used to combine them. Figure [4] shows different methods to combine the features. It was decided not to use irrelevant questions in this study, because in a Controlled Question Polygraph Test comparison between the responses to relevant and control questions is the most important factor. For most of the features seven methods were tried to combine features of different questions in a test. For the last six features three ways to combine them were tried. These methods were finding the average, maximum and minimum of relevant-control pairs. The first 93 features combined in seven ways and six integrated spectral density features were combined in three ways so the total number of features at this stage was equal to:

$$(93 \times 7) + (6 \times 3) = 669$$

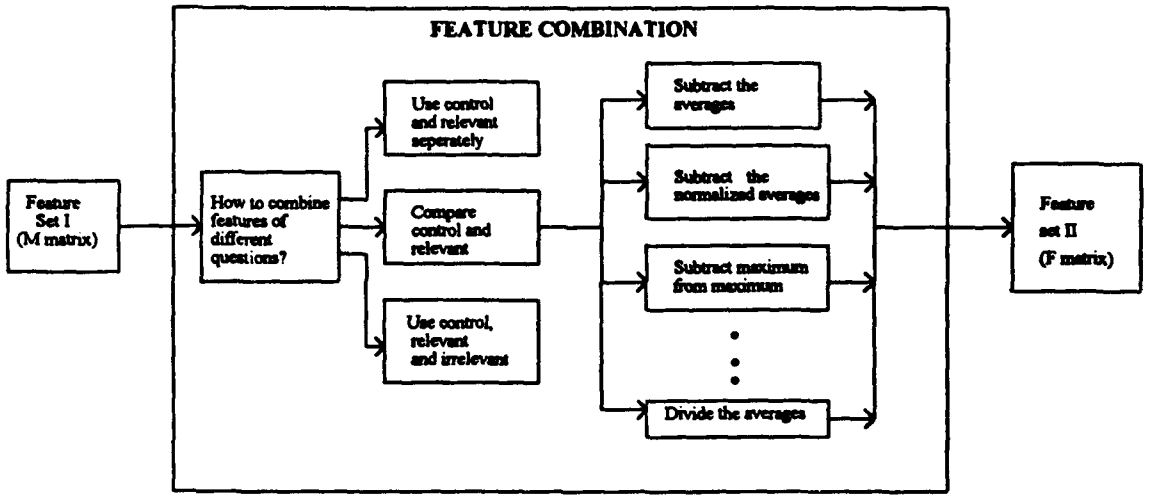


Figure 4

III.3.3 Feature Selection

Feature selection was done in two independent steps, reduction and combination. Figure [5] shows the relationship of these two steps. These two steps are explained in the following two sections.

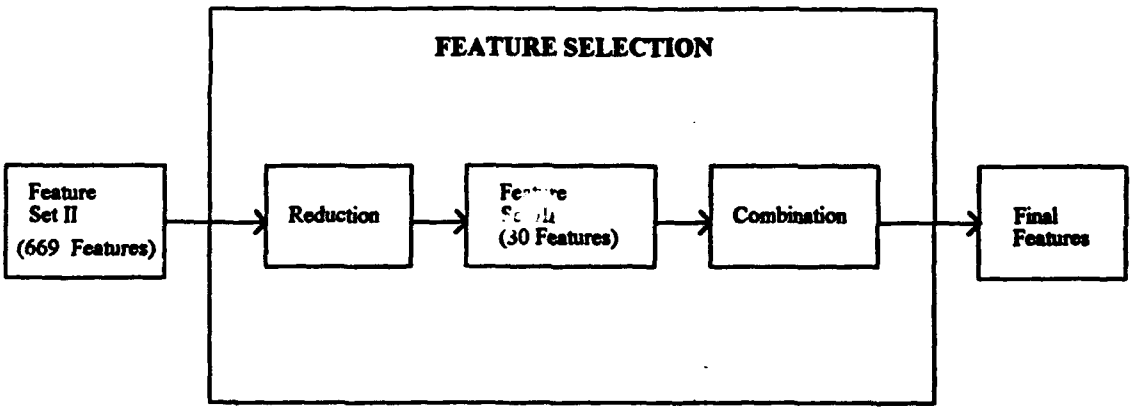


Figure. 5

III.3.3.1 Feature Selection (Reduction)

The next step in our Feature Extraction was to reduce the number of features to a number so that a practical algorithm can be used to select the feature set from them. It was decided to bring down the number of features from 669 to 30 at this step. Two different methods were chosen to test the features one at time to find the best 30. The first method was using the KNN classifier to classify the data files using one feature at a time. It was decided to use a Fuzzy version of K-nearest neighbor algorithm. The value 5 was selected for the K because it seemed that it gave better results than the other values for 1 feature classification . Also a threshold of 0.5 was used to defuzzify the output of the classifier. Refer to the section on classification for the reason of choosing this classifier. The second method was using the scatter criterion is given below.

$$J = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \qquad (1)$$

m_i = mean of class i, s_i = standard deviation of class i

This criterion measures the distance between the means of the two classes, normalized over the sum of the variances. Therefore the more compactly the samples in each class re separated, the higher will be the value of J.

The two methods were run on three sets of data. At this point a method was needed to choose the features. Different methods are possible for this step. The method that was followed is shown in figure [6] and explained below.

At first the results of KNN and scatter criterions were averaged for 3 sets of data so that features that work well for all data sets would be selected. As mentioned in an earlier section for Basic features 1 to 93, 7 features and for the features 94 to 99, 3 features were derived. Because these features are derived from one basic feature and are strongly correlated, it was decided to choose only one from them. So the best feature from these sets of 3 or 7 was selected, and the results were sorted.

Two sets of 30 features were found using the above mentioned criterions. The next step was choosing 30 features from these 60. This was done by examining the tables and selecting the features that showed a good performance in both cases or had a special physical meaning.

This set of features is the final set used for examining and selection. Table 1 in Appendix A shows these features with their corresponding meaning, channel used to derive the feature, and the method to combine the features for different questions.

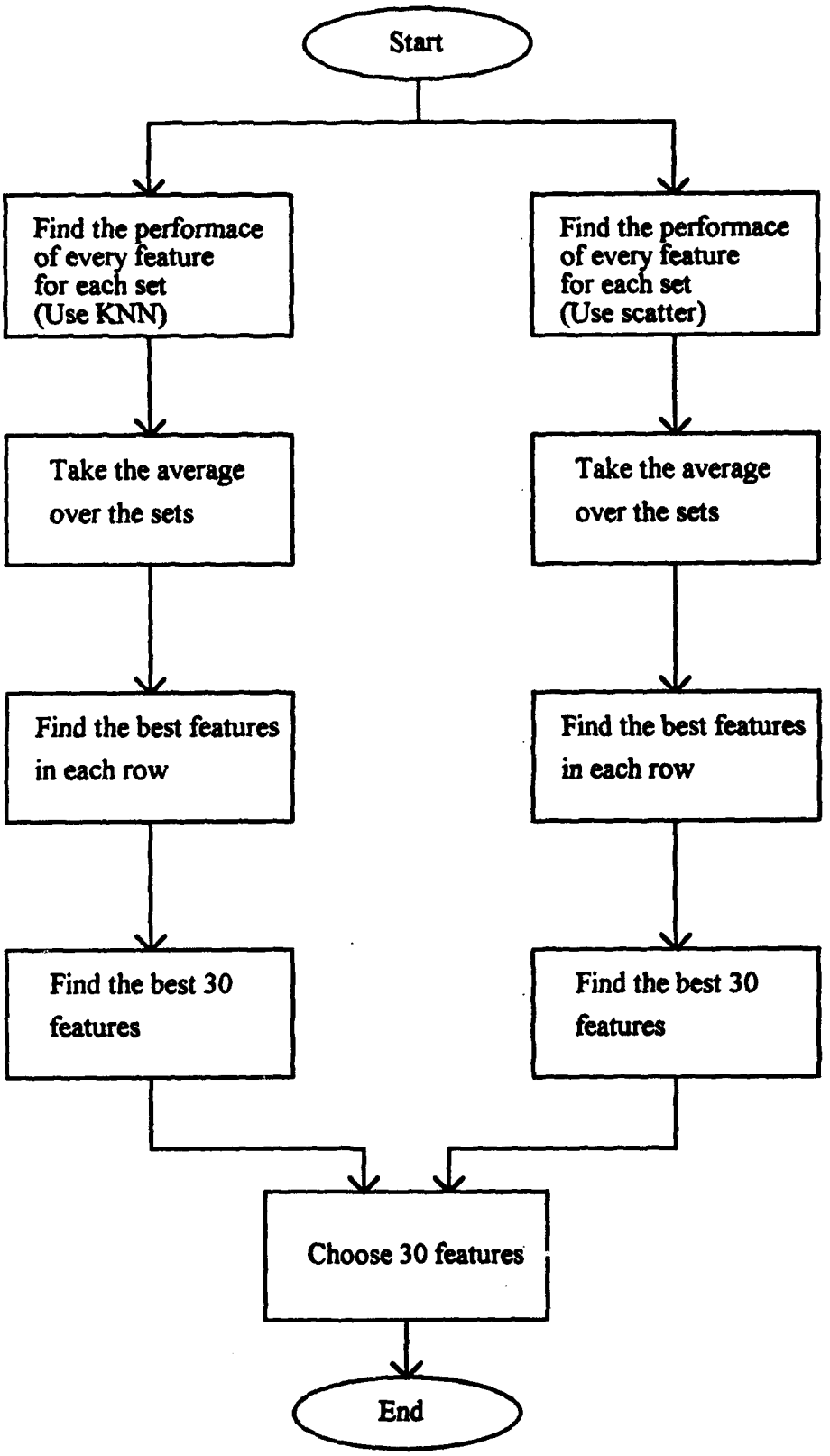


Figure. 6 Feature Selection (Reduction)

III.3.3.2 Feature Selection (Combination)

The number of features was reduced to 30 in the Feature Reduction step. This number should be further reduced because there is 100 samples in each data file, and using 30 features in a classifier might give very good results for that particular data set, but it won't be able to generalize. At this step measuring the performance of individual features is not a very logical method. Because for example features 'A' and "B' might be good features individually, but combining them might not necessarily give better results. Whereas feature 'C' that might not be a very good feature by itself might improve the classification if combined with feature 'A'.

Therefore the combinations of the features should be examined. Many methods are suggested to solve this problem. The most basic way is exhaustive search. That is trying all the combinations for these features. It is obvious that this is not practical when the number of features is not very small. For example choosing 10 or less features from a set of 30 and trying all the different combinations needs

$$\sum_{i=1}^{10}\binom{i}{30}=\sum_{i=1}^{10}\frac{30!}{i!(30-i)!}\approx 10^8$$

computations.

The method that was chosen was to start with all the combinations of two, find the best N ones among them, and use only these combinations to combine features in sets of 3. Then again find the best combinations of 3 and use them in combinations of 4 features.

This procedure is continued until satisfactory results are gained or features are not improved by increasing the number of features. Figure [7] shows the algorithm for this step.

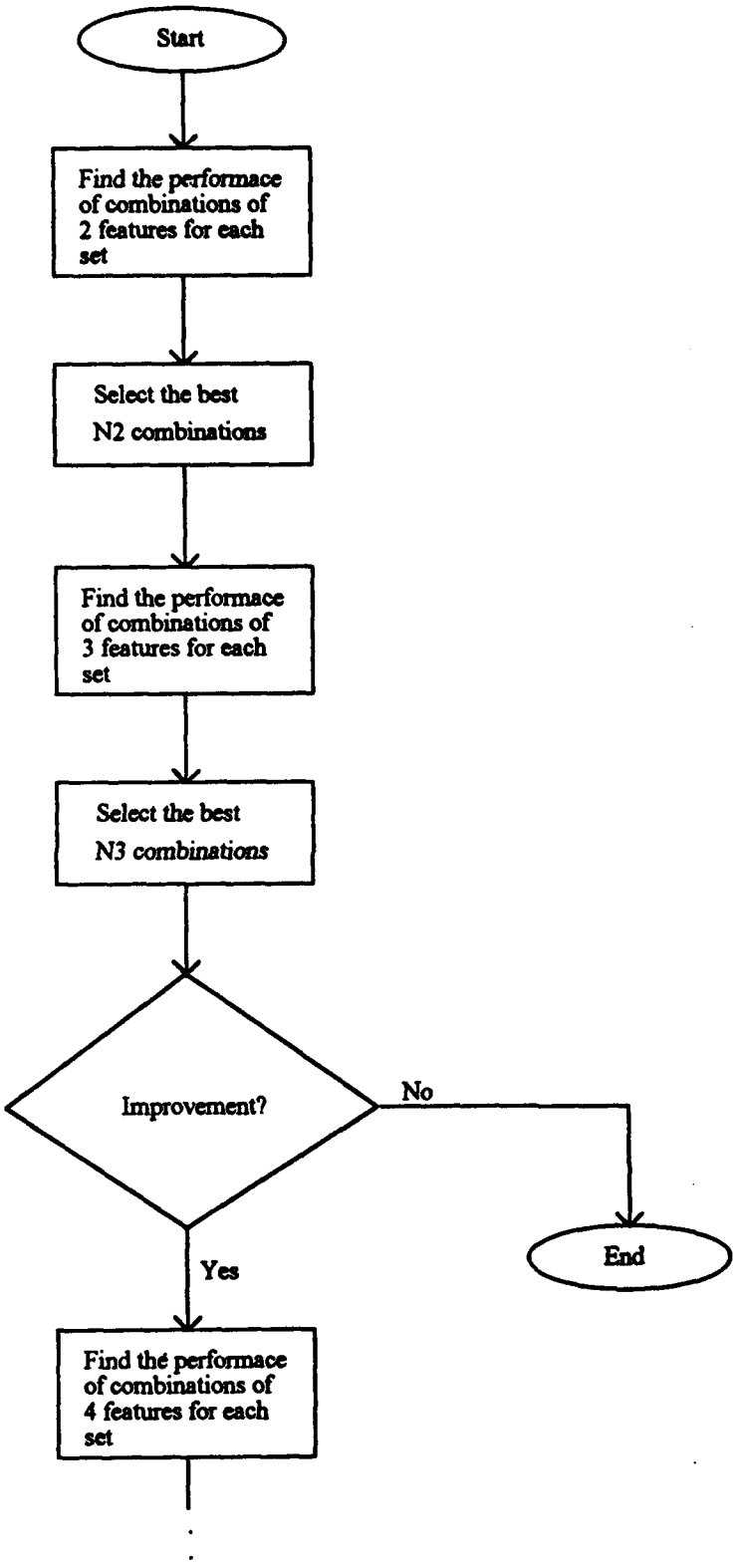


Figure 7. Feature Selection (Combination)

All pairwise combinations of the features were tried to see the classification results. The classifier used was Fuzzy K-nearest neighbor with a threshold of 0.5, and K=5. This was done for three sets of features. The results were sorted and 30 best combinations for each set were found. Also the results of classification for each combination for the 3 sets was averaged and the 30 combinations that gave best results on the average were found. These combinations are shown in Table 2 in Appendix A.

It was decided to select 20 sets of pairwise combinations to use in combinations of 3. Results for sets 1-3 and Average were studied and combinations that showed a good result in one of the sets or had a good average were selected. Table 3 in Appendix A shows these combinations.

The same steps were repeated to study the combinations of 3 and 4 features. The results are shown in Tables 4 and 6 in Appendix A. Because of time limitations it was decided not to go further from combinations of 4 features.

III.3.4 Discussion about the results:

The classification results improved consistently by increasing the number of features from one to four. The features that showed the best result for the three sets were features {5, 9, 21, 23}with 81 percent correct classification. These features represent Maximum Of GSR, Difference between Maximum and Minimum of High Cardio, Maximum of Lower Respiratory, and the Difference between Maximum and Minimum of Upper Respiratory. These features show approximately the same classification results for all three sets which is 81 percent.

Other combinations of features also gave comparable results. For example {5, 21, 23, 29} and {5, 11, 21, 23}, and {5, 10, 21, 23}. Note the repetition of {5, 21, 23}. Refer to the table 1 in Appendix A for a meaningful listing of the features. It is very notable that feature sets that show the best classification results has features that come from different channels. It can be concluded that signals from different physiological channels convey independent information, so that using features extracted from them improves the classification.

Another point to notice is that data set three shows better classification results than the two other sets, 87 percent versus 81 percent for the sets one and two. The feature set that gives the best result for data set three is {9, 14, 19, 24}. This feature set gives 87.4 percent correct classification for data set three. The feature set {5, 9, 21, 23} that gives the best classification on the average, has approximately the same results for all three sets, 81 percent. The polygraph tests that were used in this project came from several sources and were done by different examiners that used slightly different methods. Fifty consecutive tests were used to build each data set. So it is possible that some characteristic exists in the deceptive files of data set three that results in better classification. This is a matter of future investigation.

III.4. Classification

The classifier is the final stage in a pattern recognition system. The inputs to the classifier are usually a set of feature vectors. The classifier ordinarily assigns each input to one of the classes. There are many methods to design a classifier. The classifier could be designed after studying the distribution of samples of each class, or a learning classification algorithm can be implemented. We were not sure about the shape of clustering and the distribution of samples for deceptive and non deceptive classes, and it was possible that samples for one class cluster around more than one point in space. It was decided to use the K-nearest neighbor classifier* in this project because it does not explicitly use the distribution of the samples.

One of the characteristics of the conventional classification methods is that they assign each input to one of the possible classes (crisp Classification) or find probability distributions of belongingnesses of the inputs to the classes. While the way that humans think and classify objects is fundamentally different. Each object can be considered to belong to more than one class at the same time, and there are degrees of belongingness for each class. This is the basic idea that is followed in Fuzzy Logic. It was decided to follow a Fuzzy Logic based classifier in this project, because the output will be the possibility of deception and a person will not be considered completely deceptive or non deceptive.

Conventional K-nearest neighbor algorithm and a Fuzzy version of it are described in the following two sections.

* We are indebted to Professor R. Duda for suggesting KNN classifier.

III.4.1. K-Nearest Neighbor Algorithm

K-Nearest neighbor algorithm is a supervised classification method. There is no need for the training or adjusting the classifier. A set of labeled input samples is given to the classifier. When a new sample is given to the system, it finds its K nearest neighboring samples, and assigns this sample to the class that the majority of the neighbors belong to. K could be any positive integer. When K is set to 1, the algorithm is called the nearest neighbor algorithm. In this case each new sample is assigned to the class of its nearest neighbor. If K is greater than 1, it is possible that there is no majority class. To remove this tie, the sum of the distances of the new sample to its neighbors in each class is computed and the sample is assigned to the class that has the minimum distance. The main advantage of using this method is that the samples of each class are not needed to cluster in a pre specified shape. For example for a two class classification, the K-nearest neighbor classifier can still give very good results if the samples of each class are clustered in two distinct points in the space. The algorithm for the K nearest neighbor is shown in figure 8. It is supposed that C is the number of classes, K is the number of neighbors in KNN, $x_i x_i$ is the *ith* labeled sample and y is the input to be classified.

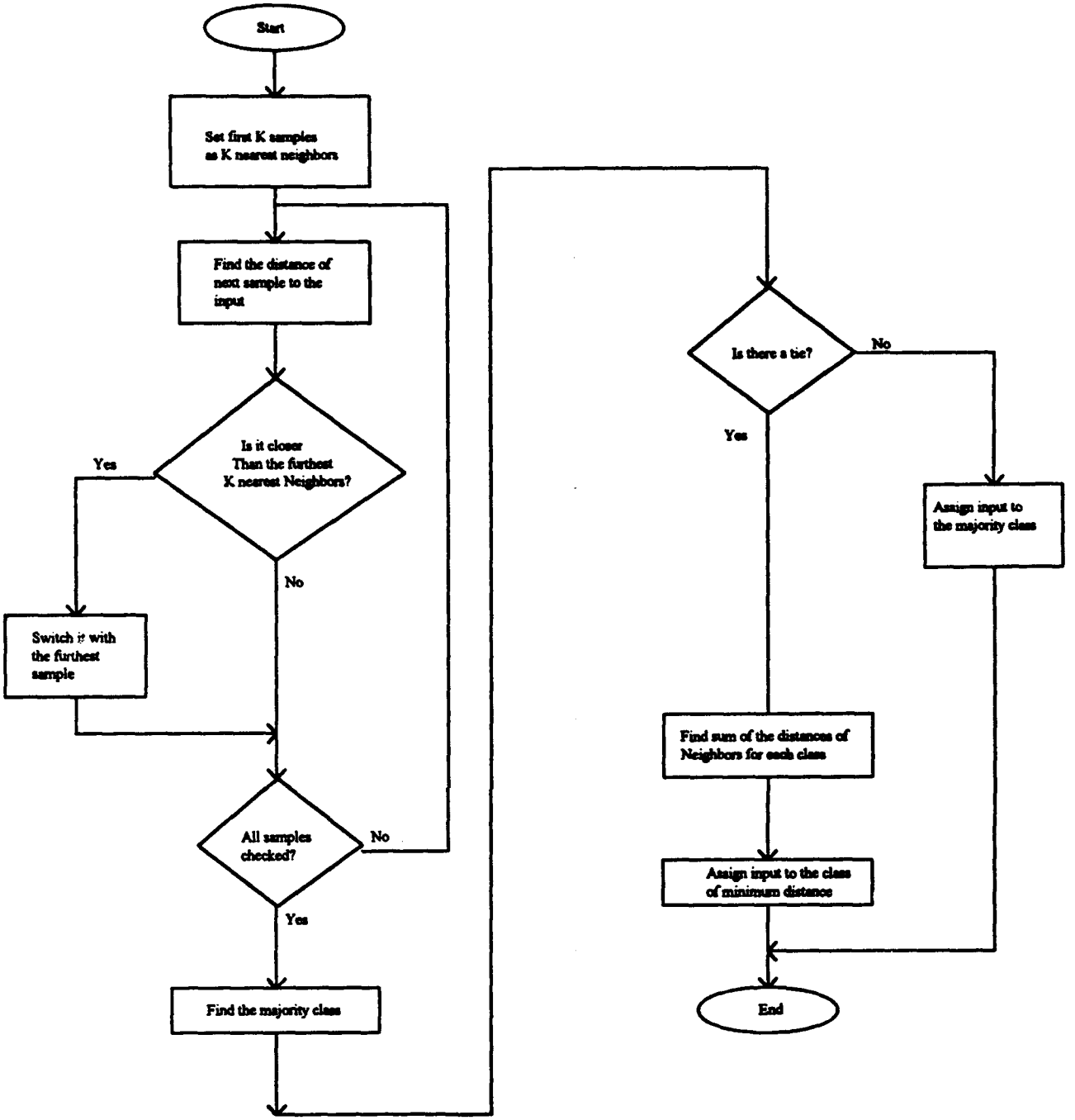


Figure 8. K Nearest Neighbor Algorithm

III.4.2. Fuzzy K Nearest Neighbor Algorithm

The fuzzy K nearest neighbor algorithm uses the same idea of conventional K nearest neighbor algorithm, that is finding the K samples that are closest to sample to be classified. But there is a conceptual difference in classification. When fuzzy classification is used, the input is not assigned to a single class. Instead, the degree of belongingness of the input to each class is determined by the classifier. By using this method more information is obtained about the input. For example if the result of classification determines membership of an input to class A is 0.9 and to class B is 0.1, it means the input belongs to class A with a very good possibility. But if the membership to class A is 0.55 and to class B is 0.45, it means that we cannot be very sure about the classification of the input. If the crisp classifier is used, in both cases the input will be assigned to class A and no further information is obtained.

Refer to [14, 15] for more detailed discussions about fuzzy K nearest neighbor algorithms. The flowchart for a fuzzy K nearest neighbor classifier is drawn in figure 9.

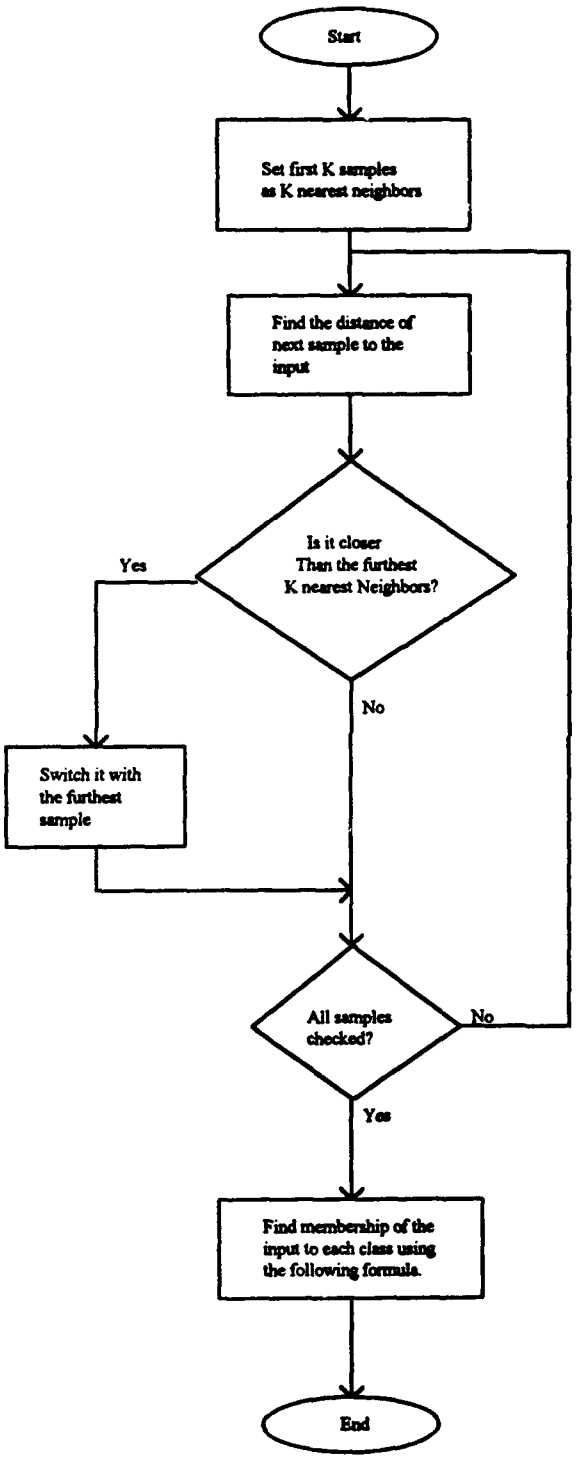
The first step in the fuzzy K nearest neighbor algorithm is the same as first step in crisp classifier. In both cases K nearest neighbors of the input are found. While in crisp classifier the majority class of the neighbors is assigned to the input, in Fuzzy classifier membership of the input to each class should be found. In order to do so the membership vector of each sample is combined to obtain the membership vector of the input. If the samples are crisply classified, membership vectors should be assigned to them. One method to do so is to assign the membership of 1 to the class that it belongs to, and membership of 0 to other classes. Other methods assign different memberships to the samples according to its distance from the mean of the class, or the distances from the nearby samples of its own class and the other classes.

When the membership vectors of the labeled samples are specified, they are combined to find the membership vector of the unknown class. This procedure should be done in a way that samples that are closer to the input have more effect on the resultant membership function. The following formula uses the inverse distance to weigh the membership

functions. x is the input to be classified, x_j is the j*th* nearest neighbor and u_{ij} is the membership of the j*th* nearest neighbor of the input in class i. D(x,y) is a distance measure between the vectors x and y which could be the Euclidean distance.

$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}(1/D(x,x_j)^{\frac{1}{m-1}})}{\sum_{j=1}^{K}(1/D(x,x_j)^{\frac{1}{m-1}})}$$

m is a parameter that changes the weighing effect of the distance. When $m >> 1$, all the samples will have the same weight. When m approaches 1, the nearest samples have much more effect on the membership value of the input.



$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}(1/D(x,x_j)^{\frac{1}{m-1}})}{\sum_{j=1}^{K}(1/D(x,x_j)^{\frac{1}{m-1}})}$$

Figure 9. Fuzzy K-Nearest Neighbor Algorithm

III.4.3. Methods and Discussion:

As mentioned in an earlier section the classifier was needed to compare the effectiveness of single features and to choose the combinations of the features that gave the best classification results. Therefore, the classifier was selected and used before the final feature set was determined. The classifier might change the results of the classification and finding the best classifier is not a trivial task. For example using the value of 10 for K may change the set of 30 best features that was found by using K = 5.

It is not practical to try all different cases for different classifiers and different parameters of classifiers, so it was decided to use a classifier with fixed parameters up to the point that final set of features were selected. The classifier as mentioned earlier was a Fuzzy K-nearest neighbor with the following parameters:

K = 5,

m = 2,

Defuzzification threshold = 0.5;

It should be noted that in order to save computation time throughout this project, each set of files was randomly broken into a training and a testing set. Each file in the testing set was classified using the labeled files in training set. Each experiment was repeated 20 times, and the results were averaged. The number of files that were used for training and testing were accordingly 75 and 25. In the last stage of experiments after the final feature set had been fixed, instead of randomly selecting testing and training files, one file was kept for testing each time and the experiment was repeated 100 times changing the test file.

After the final feature set was selected (Refer to the section on Feature Extraction), different values for K were tried on fuzzy and crisp classifier to compare the two classifiers and find the best parameters. In addition to percentage of correct classification a measure of performance was also used which is explained below.

The measure that is used to compare the performance of fuzzy classifier is the root mean square of the distances between the output of the classifier and the correct class. The correct ouput of the classifer should be 0 for non-deceptive cases and 1 for the deceptive

ones. For example if for a deceptive sample the classifier output is 0.8, 0.2 is the distance between the output and the correct class. The same measure is used for the crisp classifier. In the case of the crisp classifier the distance is always 0 for correct classification and 1 for incorrect classification.

For the fuzzy classifier the threshold used for defuzzification was also changed to find the optimum value. Tables 7 and 8 in Appendix A show the results. The best classification on the average over three sets is obtained using the fuzzy classifier with K = 6, and threshold = 0.6 . Using this values correct classification of 81.6 percent was achieved. The best result using the crisp classifier was 80.6 percent which was obtained using K=6. The performance measures for the fuzzy and crisp classifiers were accordingly 0.3915 and 0.4377 which shows fuzzy classifier has a better performance in this respect.

One final experiment that was done is explained below. In a Polygraph examination a set of questions is repeated one to five times and the decision is made by considering the responses to all these charts. In this project each chart was classified separately. As the final experiment responses to all the charts in a Polygraph examination were combined and classified as deceptive or non-deceptive. The way they were combined was finding the majority class and assigning the case to that class. In the case that equal number of files classified as deceptive and non-deceptive, the membership function of the files was averaged and the case was classified according to this value. The classification results for all the files in sets 1 to 3 are shown in Table 9 in Appendix A. The number of cases in each set was 35. The number of misclassified cases in sets 1 to 3 are 5, 7, and 3, which correspond to correct classifications of 85.7, 80.0, and 91.4 percent.

IV. Conclusion and future work

The set of four features that showed best classification results in this project were Maximum of GSR, Upper Respiration and Lower respiration signals, and the difference between the Maximum and Minimum of High Cardio signal. These are all very simple time domain features. The best classification was obtained using the fuzzy classifier with K = 6, and threshold = 0.6 . Using this values correct classification of 81.6 percent was achieved. By combining all the files in a Polygraph examination 85.7 percent correct classification was achieved on the average.

There are several suggestions for the future work. First is to repeat this work with larger sets of data files and observe the generalizability of the feature sets obtained in this research. A possible way to improve the results is to change time frames used to extract each feature for every question. In this way the optimum time for obtaining a response could be found. Another suggestion is to try different methods for fuzzification and defuzzification of feature vectors to optimize the fuzzy classifier.

REFERENCES

[1] Dale E. Olsen, et. al., "Recent developments in polygraph testing: A research review and evaluation - A technical memorandum, " Washington DC: US Government Printing Office 1983.

[2] John C. Kircher and David C. Raskin, "Human versus computerized evaluations of polygraph data in a laboratory setting, " Journal of Applied Psycology, Vol.73, 1988 No 2, pp. 291-308

[3] John E. Reid and Fred E. Inbau, Truth and Deception: The Polygraph (Lie Detector) Technique, The Williams & Wilkins Company, Baltimore, Md., 1966

[4] Michael H. Capps and Norman Ansley, "Numerical Scoring of Polygraph Charts: What Examiners Really Do", Polygraph, 1992, 21, 264-320

[5] L. A. Zadeh, "Fuzzy sets", Information and Control, vol. 8, pp. 338-332, 1965

[6] James C. Bezdek and Sankar K. Pal, Fuzzy Models for Pattern Recognition Methods that Search for Structures in Data, IEEE Press, Piscataway, NJ. 1992

[7] L. A. Zadeh, "Calculus of fuzzy restrictions," in: L. A. Zadeh, K. S. Fu, K. Tanaka and M. Shimura, eds., Fuzzy Sets and Their Applications to Cognitive and Decision Processes, Academic Press, New York, 1975, pp. 1-39

[8] Bart Kosko, Neural Networks and Fuzzy Systems, New Jersey : Prentice-Hall, Inc., 1992.

[9] Brian M. Duston, " Statistical Techniques for Classifying Polygraph Data ", Draft, November 24, 1992

[10] Howard W. Timm, " Analyzing Deception From Respiration Patterns " , Journal of Police Science and Administration, 1982, 1, 47 - 51.

[11] Personal communication with Richard Petty (polygraph examiner), June 1993

[12] Personal communication with Christopher B. Pounds (University of Washington), May 1993

[13] Personal communication with Howard Timm, May 1993

[14] J.M. Keller, M.R. Gray and J.A. Givens, "A Fuzzy K Nearest Neighbor Algorithm", IEEE Trans. on Syst. Man. Cybernetics, vol SMC-15, no. 4

[15] J.C. Bezdek and Siew K. Chuah, "Generalized K-Nearest Neighbor Rules, Fuzzy Sets and Systems vol. 18 (1986)

[16] Mitra Dastmalchi, "Feature Analysis of the Polygraph", Master's Project, San Jose State University, December 1993

[17] Eric Jacobs, "Time Domain Feature Extraction of the Polygraph", Master's Project, San Jose State University, December 1993

Appendices

Appendix A:
Tables

No.	feature	Description	Channel	Method
1	10mean	mean	GSR	1
2	10curve	curve length	GSR	2
3	10med_dif	median of the derivative	GSR	1
4	10max_min	minimum subtracted from the maximum	GSR	2
5	10max	maximum of the signal	GSR	1
6	10mdif	mean of derivative	GSR	3
7	20curve	curve length	High Cardio	1
8	20ampcard	amplitude of the peaks	High Cardio	1
9	20max_min	minimum subtracted from the maximum	High Cardio	4
10	20max	maximum of the signal	High Cardio	4
11	20min	minimum of the signal	High Cardio	1
12	30med_dif	median of the derivative	Low Cardio	3
13	30max	maximum of the signal	Low Cardio	1
14	40mean	mean	Derivative of Low Cardio	1
15	40max	maximum of the signal	Derivative of Low Cardio	1
16	50curve	curve length	Lower Respiratory	6
17	50ampr	amplitude of the peaks	Lower Respiratory	2
18	50peaknumr	number of the peaks	Lower Respiratory	5
19	50ie	inhalation divided by exhalation	Lower Respiratory	5
20	50max_min	minimum subtracted from the maximum	Lower Respiratory	2
21	50max	maximum of the signal	Lower Respiratory	6
22	60max_min	minimum subtracted from the maximum	Upper Respiratory	2
23	60max	maximum	Upper Respiratory	3
24	10std	standard deviation	GSR	2
25	20std	standard deviation	High Cardio	1
26	50std	standard deviation	Upper Respiratory	6
27	20armod1	auto regressive parameter	High Cardio	7
28	26psdcoh1	max cross spectral density	High Cardio, Lower Respiratory	1
29	10isd1	frequency of maximum integrated spectral difference of control-relevant pair	GSR	1*
30	20isd1	area under integrated spectral difference	High Cardio	3*

Methods: 1=Difference of Averages, 2=Normalized Average, 3=Max-Max, 4=Min-Min, 5=Max-Min, 6=Min-Max, 7=Max/Min , 1*=Average of relevant-control pairs, 3*=Max of relevant-control pair.

Table 1. Selected Features

Percentage of correct classification for 30 best combinations in set 1

Percent correct	Feature 1	Feature 2
74.2000	8.0000	18.0000
74.0000	10.0000	21.0000
73.0000	5.0000	7.0000
72.0000	24.0000	26.0000
71.8000	23.0000	24.0000
71.6000	4.0000	26.0000
70.4000	25.0000	26.0000
70.4000	18.0000	25.0000
70.2000	24.0000	27.0000
70.2000	9.0000	21.0000
70.0000	5.0000	27.0000
69.6000	11.0000	21.0000
69.6000	9.0000	24.0000
69.4000	11.0000	27.0000
69.4000	5.0000	26.0000
69.2000	8.0000	19.0000
69.2000	5.0000	18.0000
69.0000	25.0000	27.0000
69.0000	9.0000	18.0000
69.0000	5.0000	23.0000
68.8000	24.0000	30.0000
68.8000	18.0000	20.0000
68.8000	17.0000	20.0000
68.8000	4.0000	15.0000
68.6000	22.0000	24.0000
68.4000	6.0000	24.0000
68.4000	1.0000	27.0000
68.2000	15.0000	24.0000
68.2000	9.0000	26.0000
68.2000	5.0000	19.0000

Table [2.1] Results of pairwise combinations of features

Percentage of correct classification for 30 best combinations in set 2

Percent correct	Feature 1	Feature 2
74.4000	5.0000	23.0000
74.4000	4.0000	27.0000
74.2000	4.0000	15.0000
74.0000	20.0000	24.0000
73.6000	16.0000	24.0000
73.2000	3.0000	27.0000
72.8000	27.0000	30.0000
72.6000	4.0000	30.0000
72.6000	4.0000	7.0000
72.4000	5.0000	25.0000
72.2000	24.0000	30.0000
72.2000	8.0000	27.0000
72.2000	4.0000	17.0000
72.2000	4.0000	16.0000
72.0000	24.0000	27.0000
72.0000	24.0000	25.0000
72.0000	4.0000	20.0000
71.8000	7.0000	23.0000
71.8000	4.0000	10.0000
71.2000	25.0000	27.0000
70.8000	24.0000	26.0000
70.8000	8.0000	22.0000
70.6000	7.0000	27.0000
70.6000	6.0000	27.0000
70.4000	14.0000	21.0000
70.4000	14.0000	20.0000
70.4000	4.0000	8.0000
70.2000	4.0000	24.0000
70.0000	22.0000	27.0000
70.0000	17.0000	24.0000

Table [2.2] Results of pairwise combinations of features

Percentage of correct classification for 30 best combinations in set 3

Percent correct	Feature 1	Feature 2
81.0000	1.0000	10.0000
80.6000	9.0000	24.0000
80.4000	10.0000	24.0000
80.4000	4.0000	25.0000
80.2000	4.0000	9.0000
79.8000	5.0000	11.0000
79.2000	17.0000	24.0000
79.2000	1.0000	21.0000
79.2000	1.0000	8.0000
79.0000	1.0000	24.0000
79.0000	1.0000	11.0000
78.8000	4.0000	11.0000
78.6000	4.0000	17.0000
78.2000	24.0000	25.0000
78.2000	1.0000	14.0000
78.0000	1.0000	23.0000
78.0000	1.0000	20.0000
77.8000	23.0000	24.0000
77.8000	1.0000	5.0000
77.6000	19.0000	24.0000
77.4000	11.0000	24.0000
77.4000	5.0000	18.0000
77.2000	4.0000	19.0000
77.0000	4.0000	18.0000
76.8000	4.0000	15.0000
76.6000	5.0000	13.0000
76.6000	4.0000	24.0000
76.2000	4.0000	5.0000
76.2000	1.0000	26.0000

Table [2.3] Results of pairwise combinations of features

Percentage of correct classification for 30 best combinations in average

Percent correct	Feature 1	Feature 2
73.2667	4.0000	15.0000
72.8000	24.0000	26.0000
72.6667	4.0000	17.0000
72.6000	5.0000	23.0000
72.2667	23.0000	24.0000
72.0667	24.0000	30.0000
71.9333	20.0000	24.0000
71.8667	24.0000	27.0000
71.4667	24.0000	25.0000
71.4000	4.0000	26.0000
71.0667	4.0000	10.0000
70.9333	1.0000	8.0000
70.9333	4.0000	23.0000
70.6000	5.0000	11.0000
70.6000	4.0000	24.0000
70.5333	9.0000	24.0000
70.4667	6.0000	24.0000
70.4667	4.0000	25.0000
70.4667	4.0000	19.0000
70.4000	4.0000	30.0000
70.3333	1.0000	23.0000
70.0667	17.0000	24.0000
70.0667	1.0000	24.0000
70.0000	16.0000	24.0000
69.9333	4.0000	9.0000
69.8667	4.0000	20.0000
69.8667	5.0000	7.0000
69.8667	4.0000	7.0000
69.8000	15.0000	24.0000
69.8000	1.0000	21.0000

Table [2.4] Results of pairwise combinations of features

4	15
24	26
4	17
5	3
23	24
24	30
20	24
24	27
2[illegible]	25
4	26
1	10
9	24
10	24
5	11
17	24
4	27
16	24
8	18
10	21
5	7

Table [3]. 20 combinations of 2 features selected to combine in sets of 3

Percentage of correct classification for 30 best combinations in set 1

Percent correct	Feature 1	Feature 2	Feature 3
79.4000	10.0000	21.0000	26.0000
77.6000	5.0000	7.0000	23.0000
77.6000	5.0000	23.0000	11.0000
77.4000	5.0000	23.0000	21.0000
76.4000	16.0000	24.0000	18.0000
76.4000	5.0000	23.0000	19.0000
75.8000	23.0000	24.0000	19.0000
75.8000	23.0000	24.0000	15.0000
75.8000	5.0000	23.0000	7.0000
75.6000	5.0000	7.0000	22.0000
75.6000	5.0000	7.0000	21.0000
75.6000	5.0000	7.0000	16.0000
75.4000	5.0000	7.0000	14.0000
75.4000	5.0000	11.0000	10.0000
75.2000	10.0000	21.0000	19.0000
75.2000	8.0000	18.0000	6.0000
75.2000	5.0000	23.0000	2.0000
75.0000	10.0000	21.0000	16.0000
75.0000	10.0000	21.0000	8.0000
75.0000	5.0000	11.0000	18.0000
75.0000	4.0000	26.0000	14.0000
75.0000	5.0000	23.0000	29.0000
75.0000	5.0000	23.0000	25.0000
74.8000	10.0000	21.0000	9.0000
74.6000	10.0000	21.0000	12.0000
74.6000	5.0000	11.0000	23.0000
74.6000	10.0000	24.0000	9.0000
74.6000	5.0000	23.0000	10.0000
74.6000	5.0000	23.0000	9.0000
74.4000	5.0000	7.0000	19.0000

Table [4.1] Results of combinations of 3 features

Percentage of correct classification for 30 best combinations in set 2

Percent correct	Feature 1	Feature 2	Feature 3
79.8000	20.0000	24.0000	12.0000
78.6000	24.0000	30.0000	19.0000
78.6000	4.0000	15.0000	28.0000
78.0000	24.0000	27.0000	19.0000
77.8000	4.0000	17.0000	19.0000
77.6000	8.0000	18.0000	4.0000
77.4000	4.0000	27.0000	19.0000
77.4000	5.0000	23.0000	21.0000
77.2000	5.0000	23.0000	29.0000
77.2000	4.0000	15.0000	27.0000
77.0000	4.0000	27.0000	18.0000
77.0000	4.0000	15.0000	21.0000
76.6000	5.0000	7.0000	23.0000
76.6000	20.0000	24.0000	3.0000
76.4000	16.0000	24.0000	30.0000
76.4000	4.0000	27.0000	25.0000
76.4000	24.0000	27.0000	10.0000
76.4000	23.0000	24.0000	30.0000
76.2000	5.0000	23.0000	3.0000
76.2000	4.0000	17.0000	2.0000
76.2000	4.0000	15.0000	26.0000
75.8000	5.0000	7.0000	15.0000
75.8000	24.0000	30.0000	4.0000
75.8000	5.0000	23.0000	28.0000
75.6000	4.0000	27.0000	15.0000
75.6000	24.0000	27.0000	26.0000
75.6000	24.0000	27.0000	1.0000
75.6000	20.0000	24.0000	25.0000
75.6000	24.0000	30.0000	16.0000
75.4000	4.0000	15.0000	8.0000

Table [4.2] Results of combinations of 3 features

Appendices

Appendix A:
Tables

No.	feature	Description	Channel	Method
1	10mean	mean	GSR	1
2	10curve	curve length	GSR	2
3	10med_dif	median of the derivative	GSR	1
4	10max_min	minimum subtracted from the maximum	GSR	2
5	10max	maximum of the signal	GSR	1
6	10mdif	mean of derivative	GSR	3
7	20curve	curve length	High Cardio	1
8	20ampcard	amplitude of the peaks	High Cardio	1
9	20max_min	minimum subtracted from the maximum	High Cardio	4
10	20max	maximum of the signal	High Cardio	4
11	20min	minimum of the signal	High Cardio	1
12	30med_dif	median of the derivative	Low Cardio	3
13	30max	maximum of the signal	Low Cardio	1
14	40mean	mean	Derivative of Low Cardio	1
15	40max	maximum of the signal	Derivative of Low Cardio	1
16	50curve	curve length	Lower Respiratory	6
17	50ampr	amplitude of the peaks	Lower Respiratory	2
18	50peaknumr	number of the peaks	Lower Respiratory	5
19	50ie	inhalation divided by exhalation	Lower Respiratory	5
20	50max_min	minimum subtracted from the maximum	Lower Respiratory	2
21	50max	maximum of the signal	Lower Respiratory	6
22	60max_min	minimum subtracted from the maximum	Upper Respiratory	2
23	60max	maximum	Upper Respiratory	3
24	10std	standard deviation	GSR	2
25	20std	standard deviation	High Cardio	1
26	50std	standard deviation	Upper Respiratory	6
27	20armod1	auto regressive parameter	High Cardio	7
28	26psdcoh1	max cross spectral density	High Cardio, Lower Respiratory	1
29	10isd1	frequency of maximum integrated spectral difference of control-relevant pair	GSR	1*
30	20isd1	area under integrated spectral difference	High Cardio	3*

Methods: 1=Difference of Averages, 2=Normalized Average, 3=Max-Max, 4=Min-Min, 5=Max-Min, 6=Min-Max, 7=Max/Min , 1*=Average of relevant-control pairs, 3*=Max of relevant-control pair.

Table 1. Selected Features

Percentage of correct classification for 30 best combinations in set 1

Percent correct	Feature 1	Feature 2
74.2000	8.0000	18.0000
74.0000	10.0000	21.0000
73.0000	5.0000	7.0000
72.0000	24.0000	26.0000
71.8000	23.0000	24.0000
71.6000	4.0000	26.0000
70.4000	25.0000	26.0000
70.4000	18.0000	25.0000
70.2000	24.0000	27.0000
70.2000	9.0000	21.0000
70.0000	5.0000	27.0000
69.6000	11.0000	21.0000
69.6000	9.0000	24.0000
69.4000	11.0000	27.0000
69.4000	5.0000	26.0000
69.2000	8.0000	19.0000
69.2000	5.0000	18.0000
69.0000	25.0000	27.0000
69.0000	9.0000	18.0000
69.0000	5.0000	23.0000
68.8000	24.0000	30.0000
68.8000	18.0000	20.0000
68.8000	17.0000	20.0000
68.8000	4.0000	15.0000
68.6000	22.0000	24.0000
68.4000	6.0000	24.0000
68.4000	1.0000	27.0000
68.2000	15.0000	24.0000
68.2000	9.0000	26.0000
68.2000	5.0000	19.0000

Table [2.1] Results of pairwise combinations of features

Percentage of correct classification for 30 best combinations in set 2

Percent correct	Feature 1	Feature 2
74.4000	5.0000	23.0000
74.4000	4.0000	27.0000
74.2000	4.0000	15.0000
74.0000	20.0000	24.0000
73.6000	16.0000	24.0000
73.2000	3.0000	27.0000
72.8000	27.0000	30.0000
72.6000	4.0000	30.0000
72.6000	4.0000	7.0000
72.4000	5.0000	25.0000
72.2000	24.0000	30.0000
72.2000	8.0000	27.0000
72.2000	4.0000	17.0000
72.2000	4.0000	16.0000
72.0000	24.0000	27.0000
72.0000	24.0000	25.0000
72.0000	4.0000	20.0000
71.8000	7.0000	23.0000
71.8000	4.0000	10.0000
71.2000	25.0000	27.0000
70.8000	24.0000	26.0000
70.8000	8.0000	22.0000
70.6000	7.0000	27.0000
70.6000	6.0000	27.0000
70.4000	14.0000	21.0000
70.4000	14.0000	20.0000
70.4000	4.0000	8.0000
70.2000	4.0000	24.0000
70.0000	22.0000	27.0000
70.0000	17.0000	24.0000

Table [2.2] Results of pairwise combinations of features

Percentage of correct classification for 30 best combinations in set 3

Percent correct	Feature 1	Feature 2
81.0000	1.0000	10.0000
80.6000	9.0000	24.0000
80.4000	10.0000	24.0000
80.4000	4.0000	25.0000
80.2000	4.0000	9.0000
79.8000	5.0000	11.0000
79.2000	17.0000	24.0000
79.2000	1.0000	21.0000
79.2000	1.0000	8.0000
79.0000	1.0000	24.0000
79.0000	1.0000	11.0000
78.8000	4.0000	11.0000
78.6000	4.0000	17.0000
78.2000	24.0000	25.0000
78.2000	1.0000	14.0000
78.0000	1.0000	23.0000
78.0000	1.0000	20.0000
77.8000	23.0000	24.0000
77.8000	1.0000	5.0000
77.6000	19.0000	24.0000
77.4000	11.0000	24.0000
77.4000	5.0000	18.0000
77.2000	4.0000	19.0000
77.0000	4.0000	18.0000
76.8000	4.0000	15.0000
76.6000	5.0000	13.0000
76.6000	4.0000	24.0000
76.2000	4.0000	5.0000
76.2000	1.0000	26.0000

Table [2.3] Results of pairwise combinations of features

Percentage of correct classification for 30 best combinations in average

Percent correct	Feature 1	Feature 2
73.2667	4.0000	15.0000
72.8000	24.0000	26.0000
72.6667	4.0000	17.0000
72.6000	5.0000	23.0000
72.2667	23.0000	24.0000
72.0667	24.0000	30.0000
71.9333	20.0000	24.0000
71.8667	24.0000	27.0000
71.4667	24.0000	25.0000
71.4000	4.0000	26.0000
71.0667	4.0000	10.0000
70.9333	1.0000	8.0000
70.9333	4.0000	23.0000
70.6000	5.0000	11.0000
70.6000	4.0000	24.0000
70.5333	9.0000	24.0000
70.4667	6.0000	24.0000
70.4667	4.0000	25.0000
70.4667	4.0000	19.0000
70.4000	4.0000	30.0000
70.3333	1.0000	23.0000
70.0667	17.0000	24.0000
70.0667	1.0000	24.0000
70.0000	16.0000	24.0000
69.9333	4.0000	9.0000
69.8667	4.0000	20.0000
69.8667	5.0000	7.0000
69.8667	4.0000	7.0000
69.8000	15.0000	24.0000
69.8000	1.0000	21.0000

Table [2.4] Results of pairwise combinations of features

4	15
24	26
4	17
5	3
23	24
24	30
20	24
24	27
24	25
4	26
1	10
9	24
10	24
5	11
17	24
4	27
16	24
8	18
10	21
5	7

Table [3]. 20 combinations of 2 features selected to combine in sets of 3

Percentage of correct classification for 30 best combinations in set 1

Percent correct	Feature 1	Feature 2	Feature 3
79.4000	10.0000	21.0000	26.0000
77.6000	5.0000	7.0000	23.0000
77.6000	5.0000	23.0000	11.0000
77.4000	5.0000	23.0000	21.0000
76.4000	16.0000	24.0000	18.0000
76.4000	5.0000	23.0000	19.0000
75.8000	23.0000	24.0000	19.0000
75.8000	23.0000	24.0000	15.0000
75.8000	5.0000	23.0000	7.0000
75.6000	5.0000	7.0000	22.0000
75.6000	5.0000	7.0000	21.0000
75.6000	5.0000	7.0000	16.0000
75.4000	5.0000	7.0000	14.0000
75.4000	5.0000	11.0000	10.0000
75.2000	10.0000	21.0000	19.0000
75.2000	8.0000	18.0000	6.0000
75.2000	5.0000	23.0000	2.0000
75.0000	10.0000	21.0000	16.0000
75.0000	10.0000	21.0000	8.0000
75.0000	5.0000	11.0000	18.0000
75.0000	4.0000	26.0000	14.0000
75.0000	5.0000	23.0000	29.0000
75.0000	5.0000	23.0000	25.0000
74.8000	10.0000	21.0000	9.0000
74.6000	10.0000	21.0000	12.0000
74.6000	5.0000	11.0000	23.0000
74.6000	10.0000	24.0000	9.0000
74.6000	5.0000	23.0000	10.0000
74.6000	5.0000	23.0000	9.0000
74.4000	5.0000	7.0000	19.0000

Table [4.1] Results of combinations of 3 features

Percentage of correct classification for 30 best combinations in set 2

Percent correct	Feature 1	Feature 2	Feature 3
79.8000	20.0000	24.0000	12.0000
78.6000	24.0000	30.0000	19.0000
78.6000	4.0000	15.0000	28.0000
78.0000	24.0000	27.0000	19.0000
77.8000	4.0000	17.0000	19.0000
77.6000	8.0000	18.0000	4.0000
77.4000	4.0000	27.0000	19.0000
77.4000	5.0000	23.0000	21.0000
77.2000	5.0000	23.0000	29.0000
77.2000	4.0000	15.0000	27.0000
77.0000	4.0000	27.0000	18.0000
77.0000	4.0000	15.0000	21.0000
76.6000	5.0000	7.0000	23.0000
76.6000	20.0000	24.0000	3.0000
76.4000	16.0000	24.0000	30.0000
76.4000	4.0000	27.0000	25.0000
76.4000	24.0000	27.0000	10.0000
76.4000	23.0000	24.0000	30.0000
76.2000	5.0000	23.0000	3.0000
76.2000	4.0000	17.0000	2.0000
76.2000	4.0000	15.0000	26.0000
75.8000	5.0000	7.0000	15.0000
75.8000	24.0000	30.0000	4.0000
75.8000	5.0000	23.0000	28.0000
75.6000	4.0000	27.0000	15.0000
75.6000	24.0000	27.0000	26.0000
75.6000	24.0000	27.0000	1.0000
75.6000	20.0000	24.0000	25.0000
75.6000	24.0000	30.0000	16.0000
75.4000	4.0000	15.0000	8.0000

Table [4.2] Results of combinations of 3 features

Percentage of correct classification for 30 best combinations in set 3

Percent correct	Feature 1	Feature 2	Feature 3
85.2000	9.0000	24.0000	19.0000
85.0000	9.0000	24.0000	22.0000
84.2000	16.0000	24.0000	19.0000
84.0000	17.0000	24.0000	9.0000
84.0000	4.0000	26.0000	17.0000
83.6000	4.0000	26.0000	11.0000
83.6000	4.0000	17.0000	9.0000
83.6000	24.0000	26.0000	17.0000
83.6000	4.0000	15.0000	9.0000
83.4000	5.0000	11.0000	24.0000
83.4000	9.0000	24.0000	21.0000
83.4000	9.0000	24.0000	17.0000
83.4000	9.0000	24.0000	14.0000
83.4000	4.0000	26.0000	9.0000
83.2000	16.0000	24.0000	1.0000
83.2000	4.0000	17.0000	26.0000
83.2000	24.0000	26.0000	9.0000
83.0000	9.0000	24.0000	12.0000
83.0000	9.0000	24.0000	6.0000
83.0000	4.0000	17.0000	11.0000
82.8000	9.0000	24.0000	18.0000
82.8000	23.0000	24.0000	1.0000
82.8000	4.0000	17.0000	24.0000
82.8000	4.0000	17.0000	8.0000
82.6000	17.0000	24.0000	19.0000
82.4000	17.0000	24.0000	8.0000
82.4000	9.0000	24.0000	2.0000
82.4000	5.0000	23.0000	29.0000
82.2000	5.0000	23.0000	10.0000
82.0000	9.0000	24.0000	26.0000

Table [4.3] Results of combinations of 3 features

Percentage of correct classification for 30 best combinations on average

Percent correct	Feature 1	Feature 2	Feature 3
78.2000	5.0000	23.0000	29.0000
77.6000	5.0000	7.0000	23.0000
77.3333	5.0000	23.0000	21.0000
76.6000	5.0000	23.0000	10.0000
76.0000	23.0000	24.0000	15.0000
75.8667	5.0000	7.0000	21.0000
75.8667	5.0000	23.0000	7.0000
75.6667	5.0000	23.0000	11.0000
75.6000	8.0000	18.0000	4.0000
75.5333	4.0000	17.0000	19.0000
75.5333	5.0000	11.0000	17.0000
75.5333	24.0000	26.0000	14.0000
75.4667	5.0000	23.0000	28.0000
75.4667	4.0000	15.0000	26.0000
75.3333	17.0000	24.0000	19.0000
75.3333	5.0000	2[illegible]0000	25.0000
75.2000	5.0000	7.0000	17.0000
75.2000	4.0000	15.0000	23.0000
75.0000	5.0000	23.0000	17.0000
74.9333	5.0000	23.0000	3.0000
74.8667	4.0000	26.0000	15.0000
74.8000	23.0000	24.0000	19.0000
74.8000	5.0000	23.0000	14.0000
74.8000	5.0000	23.0000	1.0000
74.8000	24.0000	26.0000	25.0000
74.7333	24.0000	30.0000	19.0000
74.7333	5.0000	23.0000	19.0000
74.7333	5.0000	23.0000	9.0000
74.6667	5.0000	7.0000	22.0000
74.6667	4.0000	26.0000	19.0000

Table [4.4] Results of combinations of 3 features

4	17	26
5	23	29
9	19	24
4	5	9
5	10	23
5	21	23
4	8	18
19	24	30
5	7	23
19	23	24
9	14	24
4	15	28
5	11	17
4	19	17
5	23	24
5	7	21
5	11	23
14	24	26
10	21	26
4	11	26

Table [5]. 20 combinations of 3 features selected to combine in sets of 4

Percentage of correct classification for 30 best combinations in set 1

Percent correct	Feature 1	Feature 2	Feature 3	Feature 4
81.0000	5.0000	21.0000	23.0000	9.0000
80.6000	5.0000	7.0000	23.0000	6.0000
80.2000	5.0000	21.0000	23.0000	11.0000
79.6000	5.0000	21.0000	23.0000	10.0000
79.4000	5.0000	7.0000	23.0000	12.0000
79.4000	5.0000	10.0000	23.0000	21.0000
79.0000	5.0000	7.0000	23.0000	28.0000
79.0000	5.0000	7.0000	23.0000	19.0000
79.0000	5.0000	21.0000	23.0000	26.0000
78.8000	5.0000	11.0000	23.0000	7.0000
78.6000	5.0000	21.0000	23.0000	12.0000
78.4000	5.0000	21.0000	23.0000	15.0000
78.4000	5.0000	10.0000	23.0000	8.0000
78.0000	5.0000	11.0000	23.0000	21.0000
78.0000	5.0000	7.0000	23.0000	20.0000
78.0000	5.0000	7.0000	23.0000	14.0000
77.8000	5.0000	7.0000	23.0000	2.0000
77.8000	5.0000	21.0000	23.0000	28.0000
77.8000	5.0000	21.0000	23.0000	6.0000
77.8000	5.0000	21.0000	23.0000	3.0000
77.8000	5.0000	23.0000	29.0000	26.0000
77.8000	5.0000	23.0000	29.0000	22.0000
77.6000	10.0000	21.0000	26.0000	2.0000
77.6000	5.0000	7.0000	23.0000	22.0000
77.6000	5.0000	10.0000	23.0000	19.0000
77.6000	5.0000	23.0000	29.0000	19.0000
77.6000	5.0000	23.0000	29.0000	1.0000
77.4000	10.0000	21.0000	26.0000	9.0000
77.4000	5.0000	11.0000	23.0000	10.0000
77.4000	5.0000	11.0000	23.0000	8.0000

Table [6.1] Results of combinations of 4 features

Percentage of correct classification for 30 best combinations in set 2

Percent correct	Feature 1	Feature 2	Feature 3	Feature 4
81.0000	5.0000	23.0000	29.0000	14.0000
79.8000	5.0000	10.0000	23.0000	21.0000
79.6000	5.0000	21.0000	23.0000	11.0000
79.4000	14.0000	24.0000	26.0000	19.0000
79.4000	5.0000	21.0000	23.0000	9.0000
79.2000	5.0000	21.0000	23.0000	13.0000
79.0000	5.0000	11.0000	23.0000	3.0000
79.0000	5.0000	23.0000	29.0000	21.0000
78.8000	5.0000	23.0000	29.0000	6.0000
78.6000	4.0000	19.0000	17.0000	25.0000
78.6000	5.0000	21.0000	23.0000	10.0000
78.4000	4.0000	19.0000	17.0000	6.0000
78.4000	5.0000	23.0000	29.0000	19.0000
78.2000	5.0000	11.0000	23.0000	25.0000
78.2000	5.0000	11.0000	23.0000	6.0000
78.2000	4.0000	15.0000	28.0000	27.0000
78.2000	5.0000	7.0000	23.0000	11.0000
78.2000	19.0000	24.0000	30.0000	11.0000
78.0000	5.0000	21.0000	23.0000	27.0000
77.8000	19.0000	24.0000	30.0000	23.0000
77.8000	19.0000	24.0000	30.0000	16.0000
77.8000	5.0000	10.0000	23.0000	11.0000
77.6000	4.0000	19.0000	17.0000	3.0000
77.6000	5.0000	7.0000	23.0000	28.0000
77.4000	14.0000	24.0000	26.0000	20.0000
77.4000	5.0000	21.0000	23.0000	30.0000
77.2000	5.0000	11.0000	23.0000	8.0000
77.2000	4.0000	19.0000	17.0000	11.0000
77.2000	5.0000	7.0000	23.0000	26.0000
77.2000	5.0000	21.0000	23.0000	12.0000

Table [6.2] Results of combinations of 4 features

Percentage of correct classification for 30 best combinations in set 3

Percent correct	Feature 1	Feature 2	Feature 3	Feature 4
87.4000	9.0000	19.0000	24.0000	14.0000
87.2000	9.0000	14.0000	24.0000	19.0000
87.0000	9.0000	19.0000	24.0000	11.0000
86.8000	9.0000	19.0000	24.0000	18.0000
86.6000	5.0000	21.0000	23.0000	29.0000
86.6000	9.0000	19.0000	24.0000	16.0000
86.4000	9.0000	19.0000	24.0000	21.0000
86.4000	4.0000	17.0000	26.0000	18.0000
86.2000	4.0000	11.0000	26.0000	24.0000
86.2000	4.0000	8.0000	18.0000	9.0000
86.2000	9.0000	19.0000	24.0000	22.0000
86.2000	9.0000	19.0000	24.0000	6.0000
86.0000	9.0000	19.0000	24.0000	12.0000
86.0000	9.0000	19.0000	24.0000	10.0000
85.8000	9.0000	19.0000	24.0000	26.0000
85.8000	4.0000	17.0000	26.0000	9.0000
85.6000	5.0000	7.0000	21.0000	16.0000
85.6000	5.0000	7.0000	21.0000	8.0000
85.6000	9.0000	19.0000	24.0000	8.0000
85.6000	9.0000	19.0000	24.0000	5.0000
85.6000	9.0000	19.0000	24.0000	1.0000
85.4000	9.0000	14.0000	24.0000	4.0000
85.4000	5.0000	21.0000	23.0000	1.0000
85.2000	4.0000	19.0000	17.0000	10.0000
85.2000	9.0000	19.0000	24.0000	4.0000
85.0000	5.0000	11.0000	17.0000	4.0000
85.0000	9.0000	19.0000	24.0000	2.0000
85.0000	4.0000	17.0000	26.0000	8.0000
84.8000	4.0000	11.0000	26.0000	9.0000
84.8000	5.0000	21.0000	23.0000	22.0000

Table [6.3] Results of combinations of 4 features

Percentage of correct classification for 30 best combinations on average

Percent correct	Feature 1	Feature 2	Feature 3	Feature 4
81.0667	5.0000	21.0000	23.0000	9.0000
79.9333	5.0000	23.0000	29.0000	21.0000
79.8667	5.0000	21.0000	23.0000	11.0000
79.6000	5.0000	10.0000	23.0000	21.0000
79.2667	5.0000	23.0000	29.0000	19.0000
79.1333	5.0000	21.0000	23.0000	10.0000
79.0667	5.0000	23.0000	29.0000	14.0000
79.0000	14.0000	24.0000	26.0000	19.0000
78.9333	5.0000	7.0000	23.0000	12.0000
78.8667	5.0000	21.0000	23.0000	22.0000
78.8667	5.0000	7.0000	23.0000	28.0000
78.7333	5.0000	7.0000	23.0000	6.0000
78.6667	5.0000	21.0000	23.0000	7.0000
78.5333	5.0000	21.0000	23.0000	1.0000
78.4667	5.0000	23.0000	29.0000	1.0000
78.4000	5.0000	7.0000	21.0000	8.0000
78.4000	5.0000	7.0000	23.0000	26.0000
78.2667	5.0000	7.0000	23.0000	11.0000
78.2000	5.0000	7.0000	23.0000	22.0000
78.2000	5.0000	23.0000	29.0000	28.0000
78.1333	5.0000	11.0000	23.0000	10.0000
78.1333	5.0000	10.0000	23.0000	25.0000
78.0667	5.0000	7.0000	23.0000	16.0000
78.0000	5.0000	7.0000	23.0000	20.0000
77.8667	5.0000	10.0000	23.0000	29.0000

Table [6.4] Results of combinations of 4 features

k	Correct classification	Performance Index
1	73	0.5196
2	74	0.5099
3	77	0.4796
4	77	0.4796
5	82	0.42
6	81	0.4359
7	76	0.4899
8	80	0.4472
9	79	0.4583
10	79	0.4583

Table[7.1] Classification results with changing K for the crisp classifier for set 1

k	Correct classification	Performance Index
1	74	0.5099
2	74	0.5099
3	77	0.4796
4	77	0.4796
5	74	0.5099
6	76	0.4899
7	76	0.4899
8	75	0.5000
9	78	0.4690
10	78	0.4690

Table[7.2] Classification results with changing K for the crisp classifier for set 2

k	Correct classification	Performance Index
1	79	0.4583
2	79	0.4583
3	81	0.4359
4	84	0.4000
5	83	0.4123
6	85	0.3873
7	81	0.4359
8	81	0.4359
9	82	0.4243
10	82	0.4243

Table[7.3] Classification results with changing K for the crisp classifier for set 3

k	Correct classification	Performance Index
1	75.3333	0.4959
2	75.6667	0.4927
3	78.3333	0.4650
4	79.3333	0.4531
5	79.6667	0.4474
6	80.6667	0.4377
7	77.6667	0.4719
8	78.6667	0.4610
9	79.6667	0.4505
10	79.6667	0.4505

Table[7.4] Average classification results with changing K for the crisp classifier

	percent classification						performanc e index
k \ Threshold	0.3	0.4	0.5	0.6	0.7	0.8	
1	73	73	73	73	73	73	0.5196
2	77	75	73	74	72	73	0.4267
3	75	74	77	75	73	69	0.4261
4	75	74	76	77	76	69	0.4157
5	74	74	81	79	76	73	0.4061
6	69	74	78	79	76	74	0.3993
7	70	74	77	81	77	72	0.3980
8	70	75	79	79	79	72	0.3977
9	69	72	78	80	79	71	0.3971
10	68	73	78	79	79	70	0.3978

Table[8.1] Classification results for the fuzzy classifier for set 1

	percent classification						performance index
k \ Threshold	0.3	0.4	0.5	0.6	0.7	0.8	
1	74	74	74	74	74	74	0.5099
2	72	75	74	77	78	77	0.4328
3	73	75	79	79	77	73	0.4316
4	73	75	79	76	76	72	0.4262
5	71	76	76	78	77	74	0.4176
6	72	73	76	79	75	72	0.4164
7	71	73	79	79	77	70	0.4092
8	69	74	78	80	77	70	0.4099
9	73	75	80	79	77	70	0.4059
10	72	73	81	79	76	72	0.4004

Table[8.2] Classification results for the fuzzy classifier for set 2

	percent classification						performance index
k \ Threshold	0.3	0.4	0.5	0.6	0.7	0.8	
1	79	79	79	79	79	79	0.4583
2	73	76	79	84	84	84	0.3991
3	72	75	81	85	85	82	0.3862
4	75	78	84	86	86	83	0.3704
5	74	80	83	86	86	84	0.3635
6	75	82	85	87	85	83	0.3588
7	74	80	82	84	84	82	0.3605
8	73	78	83	84	84	81	0.3638
9	73	79	83	84	85	81	0.3625
10	73	80	83	84	85	82	0.3615

Table[8.3] Classification results for the fuzzy classifier for set 3

	percent classification						performanc e index
k \ Threshold	0.3	0.4	0.5	0.6	0.7	0.8	
1	75.33	75.33	75.33	75.33	75.33	75.33	0.4959
2	74	75.33	75.33	78.33	78	78	0.4195
3	73.33	74.67	79	79.67	78.33	74.67	0.4146
4	74.33	75.67	79.67	79.67	79.33	74.67	0.4041
5	73	76.67	80	81	79.67	77	0.3957
6	72	76.33	79.67	81.67	78.67	76.33	0.3915
7	71.67	75.67	79.33	81.33	79.33	74.67	0.3892
8	70.67	75.67	80	81	80	74.33	0.3905
9	71.67	75.33	80.33	81	80.33	74	0.3885
10	71	75.33	80.67	80.67	80	74.67	0.3866

Table[8.3] Average classification results with for the fuzzy classifier

File	Membership	Defuzzified	Result
1.0000	0.2736	0	
2.0000	0.3339	0	
3.0000	0.5397	0	0
4.0000	0.5450	0	
5.0000	0.7423	1.0000	
6.0000	0.1732	0	0
7.0000	0.8901	1.0000	
8.0000	1.0000	1.0000	1 Misclassified
9.0000	0.5376	0	
10.0000	0.1742	0	
11.0000	0.4366	0	0
12.0000	0.3458	0	
13.0000	0.5145	0	
14.0000	0.5178	0	0
15.0000	0.1016	0	
16.0000	0	0	
17.0000	0	0	0
18.0000	0.1334	0	0
19.0000	0	0	
20.0000	0	0	
21.0000	0.2923	0	0
22.0000	0	0	
23.0000	0	0	
24.0000	0.1607	0	0
25.0000	0	0	
26.0000	0.4421	0	
27.0000	1.0000	1.0000	0
28.0000	0.3307	0	
29.0000	0.0583	0	
30.0000	0.4965	0	0
31.0000	0.3505	0	
32.0000	0.1181	0	
33.0000	0.2101	0	0

Table [9.1] Classification of the files of set 1

File	Membership	Defuzzified	Result
34.0000	0.5970	0	
35.0000	0	0	
36.0000	0.1193	0	0
37.0000	0.3174	0	
38.0000	0.8117	1.0000	
39.0000	0.0997	0	0
40.0000	0.1889	0	
41.0000	0.4215	0	
42.0000	0.1635	0	0
43.0000	0.6474	1.0000	
44.0000	0	0	
45.0000	0.5495	0	0
46.0000	0.1115	0	0
47.0000	0	0	
48.0000	0.3986	0	
49.0000	0	0	
50.0000	0	0	0
51.0000	0.6709	1.0000	
52.0000	1.0000	1.0000	
53.0000	0.5297	0	1
54.0000	0.7245	1.0000	
55.0000	0.9200	1.0000	
56.0000	1.0000	1.0000	1
57.0000	0.9105	1.0000	
58.0000	0.9398	1.0000	
59.0000	0.5657	0	1
60.0000	0.8968	1.0000	
61.0000	1.0000	1.0000	
62.0000	0.2793	0	
63.0000	0.10[illegible]	0	0 Misclassified
64.0000	0.6245	1.0000	
65.0000	0.8643	1.0000	
66.0000	0.5054	0	1

Table [9.1] Continued

File	Membership	Defuzzified	Result
67.0000	0.8498	1.0000	
68.0000	0.6969	1.0000	
69.0000	0.8397	1.0000	1
70.0000	0.2901	0	
71.0000	0.8291	1.0000	
72.0000	0.3982	0	0 **Misclassified**
73.0000	1.0000	1.0000	
74.0000	0.2463	0	
75.0000	0.8043	1.0000	1
76.0000	0.6676	1.0000	
77.0000	1.0000	1.0000	
78.0000	1.0000	1.0000	1
79.0000	1.0000	1.0000	
80.0000	0.7538	1.0000	
81.0000	1.0000	1.0000	1
82.0000	1.0000	1.0000	
83.0000	0.8378	1.0000	
84.0000	1.0000	1.0000	1
85.0000	0.8926	1.0000	
86.0000	0.5448	0	
87.0000	0.5751	0	0 **Misclassified**
88.0000	0.8273	1.0000	
89.0000	0.2945	0	
90.0000	0.9110	1.0000	1
91.0000	1.0000	1.0000	
92.0000	1.0000	1.0000	
93.0000	0	0	1
94.0000	0.2887	0	
95.0000	0.2079	0	
96.0000	0.5793	0	0 **Misclassified**
97.0000	1.0000	1.0000	
98.0000	0.7971	1.0000	
99.0000	0.8708	1.0000	1
100.0000	1.0000	1.0000	1

Table [9.1] Continued

File	Membership	Defuzzified	Result
1.0000	0.2579	0	
2.0000	0.1307	0	
3.0000	0	0	0
4.0000	0.2652	0	
5.0000	0.4345	0	
6.0000	0.1175	0	0
7.0000	1.0000	1.0000	
8.0000	0.7086	1.0000	1 Misclassified
9.0000	0.2856	0	
10.0000	0.2745	0	
11.0000	0.3056	0	0
12.0000	0.2720	0	
13.0000	0.5019	0	
14.0000	0.8871	1.0000	0
15.0000	0.0912	0	
16.0000	0	0	
17.0000	0	0	0
18.0000	0.8334	1.0000	1 Misclassified
19.0000	0	0	
20.0000	0	0	
21.0000	0.5483	0	0
22.0000	0	0	
23.0000	0	0	
24.0000	0.1535	0	0
25.0000	0.4955	0	
26.0000	0.1013	0	
27.0000	1.0000	1.0000	0
28.0000	0.3788	0	
29.0000	0.1638	0	
30.0000	0.0905	0	0
31.0000	0	0	
32.0000	0.1431	0	
33.0000	0.0937	0	0

Table [9.2] Classification of the files of set 2

File	Membership	Defuzzified	Result
34.0000	0	0	
35.0000	0	0	
36.0000	0.1281	0	0
37.0000	0.3690	0	
38.0000	0.5734	0	
39.0000	0.1569	0	0
40.0000	0.3659	0	
41.0000	0.4124	0	
42.0000	0.1704	0	0
43.0000	0.4251	0	
44 0000	0.0664	0	
45.0000	0.5356	0	0
46.0000	0.5084	0	0
47.0000	0.1735	0	
48.0000	0.7512	1.0000	
49.0000	0.5115	0	
50.0000	0.0976	0	0
51.0000	0.6361	1.0000	
52.0000	0.8482	1.0000	1
53.0000	0.3471	0	
54.0000	0.8822	1.0000	
55.0000	1.0000	1.0000	1
56.0000	1.0000	1.0000	
57.0000	1.0000	1.0000	
58.0000	0.8730	1.0000	1
59.0000	0	0	
60.0000	0.0389	0	
61.0000	0.3643	0	0 Misclassified
62.0000	1.0000	1.0000	
63.0000	0.8174	1.0000	
64.0000	0.8875	1.0000	1
65.0000	0.7995	1.0000	
66.0000	0.5919	0	
67.0000	0.7533	1.0000	1

Table [9.2] Continued

File	Membership	Defuzzified	Result
68.0000	0.7337	1.0000	
69.0000	0.8524	1.0000	
70.0000	0.8602	1.0000	1
71.0000	0.2217	0	
72.0000	1.0000	1.0000	
73.0000	0.1268	0	0 Misclassified
74.0000	0.8860	1.0000	
75.0000	0.2121	0	
76.0000	0.1684	0	
77.0000	0.6903	1.0000	0 Misclassified
78.0000	0.7680	1.0000	
79.0000	0.8735	1.0000	
80.0000	0.8013	1.0000	1
81.0000	0.1748	0	
82.0000	0.5428	0	
83.0000	0.8496	1.0000	0 Misclassified
84.0000	0.3444	0	
85.0000	0.8298	1.0000	
86.0000	0.8590	1.0000	1
87.0000	0.6879	1.0000	
88.0000	0.9082	1.0000	
89.0000	0.6653	1.0000	1
90.0000	0.1636	0	
91.0000	0.8754	1.0000	
92.0000	0.8594	1.0000	1
93.0000	0.5185	0	
94.0000	0.4932	0	
95.0000	0.7802	1.0000	0 Misclassified
96.0000	0.8684	1.0000	
97.0000	0.8788	1.0000	
98.0000	1.0000	1.0000	1
99.0000	1.0000	1.0000	
100.0000	0.8669	1.0000	1

Table [9.2] Continued

File	Membership	Defuzzified	Result
1.0000	0.3986	0	
2.0000	0.2845	0	
3.0000	0.2562	0	0
4.0000	0.2786	0	
5.0000	0.3226	0	
6.0000	0	0	0
7.0000	1.0000	1.0000	
8.0000	0.5055	0	
9.0000	0.1434	0	0
10.0000	0	0	
11.0000	0	0	0
12.0000	0.0691	0	
13.0000	0.4744	0	
14.0000	0.4708	0	0
15.0000	0	0	
16.0000	0	0	
17.0000	0	0	0
18.0000	0.4623	0	0
19.0000	0	0	
20.0000	0	0	
21.0000	0.2096	0	0
22.0000	0	0	
23.0000	0	0	
24.0000	0.0516	0	0
25.0000	0.2885	0	
26.0000	0.0981	0	
27.0000	0.9336	1.0000	0
28.0000	0.2254	0	
29.0000	0.1465	0	
30.0000	0.0680	0	0
31.0000	0	0	
32.0000	0	0	
33.0000	0.0939	0	0

Table [9.3] Classification of the files of set 3

File	Membership	Defuzzified	Result
34.0000	0.3917	0	
35.0000	0	0	
36.0000	0	0	0
37.0000	0.1689	0	
38.0000	0.5220	0	
39.0000	0	0	0
40.0000	0.0969	0	
41.0000	0	0	
42.0000	0	0	0
43.0000	0.4810	0	
44.0000	0.3154	0	
45.0000	0.4552	0	0
46.0000	0.3285	0	0
47.0000	0.3690	0	
48.0000	0.5593	0	
49.0000	0.3522	0	
50.0000	0.2325	0	0
51.0000	1.0000	1.0000	
52.0000	0.9052	1.0000	
53.0000	0.8115	1.0000	1
54.0000	0.8397	1.0000	
55.0000	0.8754	1.0000	
56.0000	0.0930	0	1
57.0000	0.8330	1.0000	
58.0000	1.0000	1.0000	1
59.0000	1.0000	1.0000	
60.0000	1.0000	1.0000	
61.0000	1.0000	1.0000	1
62.0000	1.0000	1.0000	
63.0000	0.6496	1.0000	
64.0000	0.5075	0	1
65.0000	0.0823	0	
66.0000	0.7810	1.0000	
67.0000	0.2356	0	0 **Misclassified**

Table [9.3] Continued

File	Membership	Defuzzified	Result
68.0000	1.0000	1.0000	
69.0000	1.0000	1.0000	
70.0000	1.0000	1.0000	1
71.0000	1.0000	1.0000	
72.0000	1.0000	1.0000	
73.0000	1.0000	1.0000	1
74.0000	1.0000	1.0000	
75.0000	1.0000	1.0000	
76.0000	1.0000	1.0000	1
77.0000	1.0000	1.0000	
78.0000	1.0000	1.0000	
79.0000	1.0000	1.0000	1
80.0000	0.6068	1.0000	
81.0000	0.9054	1.0000	
82.0000	0.4134	0	1
83.0000	1.0000	1.0000	
84.0000	0	0	
85.0000	0.2914	0	0 Misclassified
86.0000	1.0000	1.0000	
87.0000	1.0000	1.0000	
88.0000	0.8786	1.0000	1
89.0000	0.9018	1.0000	
90.0000	1.0000	1.0000	
91.0000	1.0000	1.0000	1
92.0000	1.0000	1.0000	
93.0000	0.9135	1.0000	
94.0000	0.8292	1.0000	1
95.0000	0.7423	1.0000	
96.0000	1.0000	1.0000	
97.0000	0.0902	0	1
98.0000	0.2564	0	
99.0000	0	0	
100.0000	0.4387	0	0 Misclassified

Table [9.3] Continued

Non deceptive	Deceptive 1	Deceptive 2	Deceptive 3
QQ8R9OIO.011	QQ4Q1O83.011	QQ7LX5Q0.021	QQ8RAJ0C.011
QQ8R9OIO.021	QQ4Q1O83.021	QQ7LX5Q0.031	QQ8RAJ0C.021
QQ8R9OIO.031	QQ4Q1O83.031	QQ7MN2Y0.011	QQ8RAJ0C.031
QQ95LU1T.011	QQ4Q3MDC.011	QQ7MN2Y0.021	QQ9EUKVT.011
QQ95LU1T.021	QQ4Q3MDC.021	QQ7MN2Y0.031	QQ9EUKVT.021
QQ95LU1T.031	QQ4Q3MDC.031	QQ7TC5UF.011	QQ9EUKVT.031
QQAURNUS.021	QQ51DE36.011	QQ7TC5UF.021	QQ9IOOXO.021
QQAURNUS.031	QQ51DE36.021	QQ7TC5UF.031	QQ9IOOXO.041
QQAV53P6.011	QQ51DE36.041	QQ7TQVER.011	QQ9SOW8L.011
QQAV53P6.021	QQ6RQGH6.011	QQ7TQVER.021	QQ9SOW8L.021
QQAV53P6.031	QQ6RQGH6.021	QQ7TQVER.031	QQ9SOW8L.031
QQBQ4SHI.011	QQ6RQGH6.031	QQ7TVADC.011	QQ9SQIK9.011
QQBQ4SHI.021	QQ6RQGH6.041	QQ7TVADC.021	QQ9SQIK9.021
QQBQ4SHI.031	QQ6T711O.011	QQ7TVADC.031	QQ9SQIK9.031
QQBSS7WT.011	QQ6T711O.021	QQ7U2T4R.011	QQ9W0B9F.011
QQBSS7WT.021	QQ6T711O.031	QQ7U2T4R.021	QQ9W0B9F.031
QQBSS7WT.031	QQ6Z59IG.011	QQ7U2T4R.031	QQ9W0B9F.041
QQ7OXM60.021	QQ6Z59IG.021	QQ7YP7QU.011	QQ9U4FMU.011
QQ7RH0RO.011	QQ6Z59IG.031	QQ7YP7QU.021	QQ9U4FMU.021
QQ7RH0RO.021	QQ7PP9B9.011	QQ7YP7QU.031	QQ9U4FMU.031
QQ7RH0RO.031	QQ7PP9B9.021	QQ7YZOJ3.011	QQ9Y_SVF.011
QQ7R51P9.011	QQ7PP9B9.031	QQ7YZOJ3.021	QQ9Y_SVF.021
QQ7R51P9.021	QQ7PDU1X.011	QQ7YZOJ3.031	QQ9Y_SVF.031
QQ7R51P9.031	QQ7PDU1X.021	QQ8_0DPT.011	QQ9YH3QF.011
QQ9TDSP3.011	QQ7PDU1X.031	QQ8_0DPT.021	QQ9YH3QF.021
QQ9TDSP3.021	QQ7_PIPF.011	QQ8_0DPT.031	QQ9YH3QF.031
QQ9TDSP3.031	QQ7_PIPF.021	QQ8_0DPT.041	QQA2TT4C.011
QQA8OWOI.011	QQ7_PIPF.031	QQ8_2UQ9.011	QQA2TT4C.021
QQA8OWOI.021	QQ7_JT70.011	QQ8_2UQ9.021	QQA2TT4C.031
QQA8OWOI.031	QQ7_JT70.021	QQ8_2UQ9.031	QQA3HIRX.011
QQBT22O6.011	QQ7_JT70.031	QQ800IG6.011	QQA3HIRX.021
QQBT22O6.021	QQ738DYX.011	QQ800IG6.021	QQA3HIRX.031
QQBT22O6.031	QQ738DYX.021	QQ800IG6.031	QQA32UTF.011
QQBO9O_9.011	QQ738DYX.031	QQ82OIU9.011	QQA32UTF.021
QQBO9O_9.021	QQ75ULP9.011	QQ82OIU9.021	QQA32UTF.031
QQBO9O_9.031	QQ75ULP9.021	QQ82OIU9.031	QQA6U_IF.011
QQBC7PP6.011	QQ75ULP9.031	QQ82SUTX.011	QQA6U_IF.031
QQBC7PP6.021	QQ79_EYF.011	QQ82SUTX.021	QQA6U_IF.041
QQBC7PP6.031	QQ79_EYF.021	QQ82SUTX.031	QQAM4E3L.011
QQCHCK_O.011	QQ79_EYF.031	QQ860ZNU.011	QQAM4E3L.021
QQCHCK_O.021	QQ7BGDML.011	QQ860ZNU.021	QQAM4E3L.031
QQCHCK_O.031	QQ7BGDML.021	QQ860ZNU.031	QQARF2_X.011
QQCDTKP0.011	QQ7BGDML.031	QQ89U_ZR.011	QQARF2_X.021
QQCDTKP0.031	QQ7ETC8I.011	QQ89U_ZR.021	QQARF2_X.031
QQCDTKP0.041	QQ7ETC8I.021	QQ89U_ZR.031	QQAWA38X.011
QQCM5Y56.011	QQ7ETC8I.031	QQ8ATU26.011	QQAWA38X.021
QQCQQT8Y.011	QQ7JAQCS.011	QQ8ATU26.021	QQAWA38X.031
QQCQQT8Y.021	QQ7JAQCS.021	QQ8ATU26.031	QQAYXZGU.011
QQCQQT8Y.031	QQ7JAQCS.031	QQ8FGMVI.011	QQAYXZGU.021
QQCQQT8Y.041	QQ7LX5Q0.011	QQ8FGMVI.021	QQAYXZGU.031

Table [10] NSA Polygraph files used in sets 1-3.

Note: Each set consists of non-deceptive files and one of the deceptive sets

Appendix B:
Program Listings

Classify Program

```
% This is a Matlab program
% This script parses a matrix of polygraph
% vectors into training and testing vectors.
% It then calls the classifier, trains, tests
% and gives results.


c = 2;                  % number of classes
percent_train=.75;      % percentage of inputs used for training

features=[1]            % features to use
classification=1;  % use fuzzy classifier
kk=5;                   % K in K nearest neighbor
change=1;               % Randomize training and testing inputs
repeat=20;              % Number of repeatitions
ut=.5;                  % Upper threshhold for 3 class fuzzy classifier
lt=.5;                  % Lower threshhold for 3 class fuzzy classifier

load set31;               % file containing feature matrix
                              % and vector that indicates whether
                                 % column is truthful or deceptive
%classvect;                     % vector of classes eg. 1 = deceptive
                        % 0 = truthful vector
featurematrix = featmat;        % matrix of features
dimension = size(featurematrix);
columns = dimension(2);             % the total number of columns in the feature matrix
number_train = round(percent_train*columns);      % number of vectors
                                    % used for training


ur=.5;                  %upper threshold
continue=1;                     % to repeat the program
while (continue==1)
'=============================================================================='
apercent_classified=[];         % clear average results
acorrect=[];
acc=[];
ffresult=[];
ccresult=[];
ttestclass=[];

men=0;
while(men ~=7)
        men=menu('Select:','Features','Type','K','Random'...
        ,'Repeat','% training','Start','Defuzz','Exit');

        if (men==1)
        'enter a vector of the features you want tested (eg. [1 2 4])  '
```

```
        features = input(' ');          % features being tested
        end

        if (men==2)
                classification=menu('Type:','Fuzzy','Crisp');
        end

        if (men==3)
        kk = input('enter the "K" in K nearest neighbor  ')
        end

        if (men==4)
                change=menu('Selection','Random','Constant');
        end

        if (men==5)
                repeat=input('Enter number of repeatitions')
        end

        if (men==6)
                percent_train=input('Enter percentage of the files used for training, 1 for all-1')
        end
        if (men==8)
                ch=menu('Defuzzification', '3class', 'Upper thresh','Lower thresh');
                if ch==1,          classification=3,  end
                if ch==2
                        ut=input('enter the upper threshhold'); % lower limit for class 1
                end
                if ch==3
                        lt=input('enter the lower threshhold'); %upper limit for class 0
                end
        end
        if (men==9) break,end
end
if men==9 break,end
number_train = round(percent_train*columns);
acorrect=[];                % vector for the average of correct classification
acc=[];                     % vector for the average of performance index

if percent_train == 1       % To repeat nonrandom testing for all the files.
        repeat =columns;
end

for trial=1:repeat

featurematrix = featmat(features,:);   % creates a feature matrix of the
                                       % the features being tested
if ( (change==1) & (percent_train~=1) )
        [trainvect, testvect] = randvect(number_train,columns);
end;
if percent_train == 1
        testvect = trial;
        if (trial ==1)
                trainvect=2:columns;
```

```
        end
        if (trial == columns)
                trainvect=1:columns-1;
        end
        if ( trial ~=1  &  trial ~=columns )
                trainvect = [1:trial-1 , trial+1:columns];
        end
end
testvect
trainvect
u = featurematrix(:,testvect);          % testing matrix

testclass = classvect(1,testvect);      % class of each column in testing matrix

p = featurematrix(:,trainvect);         % training matrix

t = classvect(1,trainvect);             % class of each column in training matrix


if classification == 1                  % Fuzzy classifier

        % m = input('enter the degree of fuzziness "M" (1<=M<=infinfity)')
        m = 2;
        save fdatafil c kk m p t u
%       !fknn                           %This line invokes the classifier program in a dos window
        dos('del foutfile.mat|')        %to make sure that the program actulally works
        dos('fknn|')
        'Now loading the result of the fuzzy classifier'
        load foutfile
        '-------------------------------------------------'
        kk, features
        fresult
        testclass

        if(percent_train==1)
                ffresult=[ffresult fresult]
                ttestclass=[ttestclass testclass];
        end

        cr =fresult(2,:) > ut           % defuzzification of the result
        correct = 100*(1-mean(abs(testclass-cr)))  % percentage correct classified
        cc = [1-testclass; testclass];          % adding a row of complements to c
        cc=fresult-cc;
        'Performance Index='
        cc = sqrt(mean(mean(cc .^ 2)))
end

if classification == 2                  % crisp classifier

        save cdatafil c kk p t u
%       !cknn           %This line invokes the classifier program in a dos window
        dos('del foutfile.mat|')        %to make sure that the program actulally works
        dos('cknn|')
        'Loading the Crisp output file'
```

```
        load coutfile
        '-------------------------------------------------'
        kk, features
        cresult
        testclass

        if(percent_train==1)
                ccresult=[ccresult cresult]
                ttestclass=[ttestclass testclass];
        end

        correct = 100*(1-mean(abs(testclass-cresult)))  % percentage correct classified
        cc = sqrt(mean(abs(testclass-cresult)))         % performance index

end
if classification == 3                  % Fuzzy classifier but defuzzification into 3 classes

        % m = input('enter the degree of fuzziness "M" (1<=M<=infinfity)')
        m = 2;
        save fdatafil c kk m p t u
%       !fknn                           %This line invokes the classifier program in a dos window
        dos('del foutfile.mat|')        %to make sure that the program actulally works
        dos('fknn|')
        'Now loading the result of the fuzzy classifier'
        load foutfile
        '-------------------------------------------------'
        kk, features
        fresult
        testclass

        if(percent_train==1)
                ffresult=[ffresult fresult]
                ttestclass=[ttestclass testclass];
        end
        class1=find(fresult(2,:) >ut);
        class0=find(fresult(2,:) <lt);
        class3=find(fresult(2,:) >lt & fresult(2,:) <ut);
        percent_classified=100*((length(class0)+length(class1))/length(testclass))
        fr=[fresult(:,class1) fresult(:,class0)]        % the section that is classified into one of the two
classes
        cr=fr(2,:)>ut
        tr=[testclass(class1) testclass(class0)]        % the section that is classified into one of the two
classes
        correct = 100*(1-mean(abs(tr-cr)))  % percentage correct classified
        cc = [1-tr; tr];            % adding a row of complements to cc
        cc=fr-cc;
        'Performance Index='
        cc = sqrt(mean(mean(cc .^ 2)))
end

        apercent_classified = [apercent_classified percent_classified]
        acorrect=[acorrect correct]
        acc=[acc cc]
```

```
end                % for trial

if classification ==3                          % 3 class fuzzy
apercent_classified=mean(apercent_classified)
end
acorrect, mean(acorrect)
acc, mean(acc)

continue=3;
while (continue == 3 | continue==4)
continue=menu('Repeat?', 'Yes', 'no','Plot', 'threshold');
if(continue==3)
        dim=menu('Dimension', 'Two', 'Three')+1;
        if(dim==2)

                pp=p(:,find(t));
                plot(pp(1,:),pp(2,:),'r+');
                title('A clustering of two class data');
                hold on
                pp=p(:,find(t==0));
                plot(pp(1,:), pp(2,:), 'gx');

                pp=u(:, find(testclass));
                plot(pp(1,:), pp(2,:), 'r+');
                pp=u(:,find(testclass==0));
                plot(pp(1,:), pp(2,:), 'gx');

                hold off
        end     %if(dim==2)

        if(dim==3)

                pp=p(:,find(t));
                plot3(pp(1,:),pp(2,:), pp(3,.), 'r+');
                title('A clustering of two class data');
                hold on
                pp=p(:,find(t==0));
                plot3(pp(1,:), pp(2,:), pp(3,:), 'rx');

                pp=u(:, find(testclass));
                plot3(pp(1,:), pp(2,:), pp(3,:), 'g+');
                pp=u(:,find(testclass==0));
                plot3(pp(1,:), pp(2,:), pp(3,:), 'gx');

                hold off
        end     %if(dim==3)

end    %if(continue==3)

if (continue==4)

        ch=menu('Defuzzification', '3class', 'Upper thresh','Lower thresh');
        if ch==1,          classification=3,  end
```

```
        if ch==2
                ut=input('enter the upper threshhold'); % lower limit for class 1
        end
        if ch==3
                lt=input('enter the lower threshhold'); %upper limit for class 0
        end

        if classification==1
                cr =ffresult(2,:) > ut          % defuzzification of the result
                correct = 100*(1-mean(abs(ttestclass-cr)))  % percentage correct classified
                cc = [1-ttestclass; ttestclass];          % adding a row of complements to c
                cc=ffresult-cc;
                'Performance Index='
                cc = sqrt(mean(mean(cc .^ 2)))
        end

        if classification==2
                correct = 100*(1-mean(abs(ttestclass-ccresult)))  % percentage correct classified
                cc = sqrt(mean(abs(ttestclass-ccresult)))        % performance index
        end

        if classification==3
                class1=find(ffresult(2,:) >ut);
                class0=find(ffresult(2,:) <lt);
                class3=find(ffresult(2,:) >lt & ffresult(2,:) <ut);
                fr=[ffresult(:,class1) ffresult(:,class0)]        % the section that is classified into one of
the two classes
                cr=fr(2,:)>ut
                tr=[ttestclass(class1) ttestclass(class0)]        % the section that is classified into one of
the two classes
                percent_classified=100*((length(class0)+length(class1))/length(ttestclass))
                correct = 100*(1-mean(abs(tr-cr)))  % percentage correct classified
                cc = [1-tr; tr];          % adding a row of complements to cc
                cc=fr-cc;
                'Performance Index='
                cc = sqrt(mean(mean(cc .^ 2)))
        end
end

end     % while continue == 3 | 4

end                     % while continue
```

```
/*  This program implements a K-nearest neighbor classifier.
    created by: Shahab Layeghi

    created: 8/4/93
    last modified: 9/17/93

*/
```

```
/* The main program opens a matlab data file, reads the training matrix,
classifies each entry in the testing matrix, and writes the result in an
output file. The file that this program gets the information from should be
called "cdatafil.mat". As the name implies it is in matlab file format.
The data in this file should have the following order:

1. A single variable 'C' which is the number of classes.
2. A single variable 'K' which is the parameter 'K' in K-NN Algorithm.
3. A trainig matrix 'P' which contains a set of feature vectors. Each vector
is in a column of the matrix.
4. A classes vector 'T' which contains the classes of the training set
5. An input matrix 'U' which contains a set of unclassified feature vectors.

The main program uses the CrispKNN routine to classify each one of the input
vectors and saves the results (the classes that these inputs belong to) in a
file called coutfile.mat. This file is in Matlab format. This file contains
a vector of the classes called:

'cresult'


This program can be called from dos, or within Matlab by using dos escpae
character '!'. An example Matlab script file that shows how this program can
be used is included in the file "cknntest.m".

*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <conio.h>

#define INPUTFILE "cdatafil.mat"
#define OUTPUTFILE "coutfile.mat"

// Function Prototypes ----------------------------------------

int CrispKNN(double *Input, double *Samples, double *Lables);
double FindDistance(double *vec1, double *vec2);
double Maxd(double *vec, int *index, int Length);
int FindMax(int *vector, int *count, int Length, int Max);
int loadmat(FILE * fp,int *type, char *pname, int *mrows, int *ncols,
                int *imagf, double **preal, double **pimag);
void savemat(FILE *fp, int type, char *pname, int mrows, int ncols,
                int imagf, double *preal, double *pimag);


// Global variables, these variables will be set by reading matlab file ------

int classes;          /* the number of classes */
int features;         /* Number of features in a class */
```

```
int KK ;                                    /* K in K-nearest neighbors */
int SampleSize;                             /* Number of Labled Samples */
int TestSize;

//-------------------------------------------------------------


/*-------------------------------------------------------------*/

void main()
{

        double *Lables;
        double *KP;
        double *input;
        int i,j;
        FILE *fp;
        char name[20];
        int type, imagf;
        double *Samples, *isamples;    // isamples is for imaginary part of the matrix that is not used in
here.
        double *Testdata;
        double *result;
        fp=fopen(INPUTFILE,"rb");
        if(!fp) {
                printf("cannot open the file");
                exit(-1);
        }
        // read classes from the file
        loadmat(fp, &type, name, &i, &j, &imagf, &KP, &isamples);
        if(i!=1 || j!=1) {
                printf("error: You should include classes at the beginning of the file\n");
                exit(-1);
        }
        classes=*KP;

        // read KK from the file
        loadmat(fp, &type, name, &i, &j, &imagf, &KP, &isamples);
        if(i!=1 || j!=1) {
                printf("error: You should include K at the beginning of the file\n");
                exit(-1);
        }
        KK=*KP;

        // read the matrix from the datafile.
        loadmat(fp, &type, name, &features, &SampleSize, &imagf, &Samples, &isamples);

        // reading lables from data file
        loadmat(fp, &type, name, &i, &j, &imagf, &Lables, &isamples);
        if(i!=1 || j!=SampleSize) {
                printf("error: Number of labels is different from the number of samples\n");
                exit(-1);
        }
```

```
        // read data to be classified from the file
        loadmat(fp, &type, name, &i, &TestSize, &imagf, &Testdata, &isamples);
        if(i != features) {
                printf("error: Training and testing matrices should have the same size");
                exit(-1);
        }

        // Allocate space for result vector

        result = (double *) malloc(TestSize*sizeof(double));
        if(!result) {
                printf("Error: cannot allocate memory for the result vector");
                exit(-1);
        }

        for(i=0; i<TestSize; i++) {          // for each input
                input=Testdata+i*features;
                result[i]=CrispKNN(input, Samples, Lables);
//              printf("class: %lf\n", result[i]);
        }
        fclose(fp);
//      printf("\n  End of classification, Now writing the result in the file");

        fp=fopen(OUTPUTFILE, "wb");
        if(!fp) {
                printf("Error: Cannot write the file");
                getch();
        }
        savemat(fp, 0, "cresult", 1, TestSize, 0, result, result);
        fclose(fp);

}

/*-------------------------------------------------------------*/
int CrispKNN(double *Input, double *Samples, double *Lables)
{
        int i,j ;
        int nj, k, nk;
        double *distance;
        int *index;
        double x,y;

        distance = (double *) malloc(KK*sizeof(double));
        if(!distance) {
                printf("Error: Not enough memory for distance vector");
                exit(-1);
        }

        index = (int *) malloc(KK*sizeof(int));
        if(!index) {
                printf("Error: Not enough memory for index vector");
                exit(-1);
        }
```

```
        for(i=0; i<KK; i++) {  // This loop initializes K nearest neighbors to the first K Samples
                index[i]=Lables[i]+1;
                distance[i]=FindDistance(Input, &Samples[i*features]);
        }
        for(i=KK; i<SampleSize; i++) {    // This is the loop that finds the K nearest Neighbors
                x=Maxd(distance, &j, KK);
                y=FindDistance(Input, &Samples[i*features]);
                if(y < x) {  // This sample is closest to the input than the farthest K Neighbors
                        distance[j]=y;
                        index[j]=Lables[i]+1;
                }
        }
        j=FindMax(index, &nj, KK, classes);   // Finds the class of maximum occurance

        /* In this section it is checked to see if there is a tie.  That is if
        there are two or more classes with the same number of occureances.  If
        there is a tie for two classes,  the class with the minimum sum of
        distances is selected. No action is taken for a tie of more than two
        classes.  */

        for (i=0; i<KK; i++)
                if(index[i]==j) index[i]=0;
        k=FindMax(index, &nk, KK, classes);
        if(nk==nj) {                    // If there is a tie.
                x=0;
                for(i=0; i<KK; i++) {
                        if(index[i]==0)
                                x+=distance[i];
                }
                y=0;
                for(i=0; i<KK; i++) {
                        if(index[i]==k)
                                y+=distance[i];
                }
                if(y<x)                                         //If sum of the distances to class j is
less than that of class k
                        j=k;
        }

        free(distance);
        free(index);
        return j-1;

}

/*-------------------------------------------------------------------*/
/* This function returns the Euclidian distance between two vectors */

double FindDistance(double *vec1, double *vec2)
{

        int k;
        double distance;
```

```
        distance = 0;
        for(k=0; k<features; k++) {
                        distance +=(vec1[k]-vec2[k])*(vec1[k]-vec2[k]);
//                      distance += pow(vec1[k]-vec2[k] , 2);
        }
        return distance;
}

/*-----------------------------------------------------------------*/
/* This function finds the biggest element of an array.  It returns that
value and also returns the index to that element in index.
*/

double Maxd(double *vec, int *index, int Length)
{
        int i,j=0;

        j=0;
        for(i=1; i<Length; i++)
                if(vec[i]>vec[j]) j=i;
        *index=j;
        return(vec[j]);
}

/*-----------------------------------------------------------------*/
/* This function finds a number that is most often repeated in an array of
integer values, and returns that number.  Length of array shoud be less than
100.  It is supposed that number is an integer greater than zero.
vector is a pointer to the array.  count is the number of times that the
number is repeated.  Length is the length of the vector.
*/

int FindMax(int *vector, int *count, int Length, int Max)
{

        int i, j, m;
        int t[101];

        if(Max>100) Max=100;
        for(i=0; i<Max+1; i++)
                t[i]=0;
        for(i=0; i<Length; i++)
                t[vector[i]]++;
        m=t[1];
        j=1;
        for(i=1; i<Max+1; i++) {
                if(t[i]>m) {
                        m=t[i];
                        j=i;
                }
        }
        *count=m;
        return (j); }
```

```
/*      This program implements a fuzzy version of K-nearest neighbor classifier.
        created by: Shahab Layeghi

        created: 9/1/93
        last modified: 9/3/93

*/

/* The main program opens a matlab data file, reads the training matrix,
classifies each entry in the testing matrix, and writes the result in an
output file. The file that this program gets the information from should be
called "fdatafile.mat". As the name implies it is in matlab file format.
The data in this file should have the following order:

1. A single variable 'C' which is the number of classes.
2. A single variable 'K' which is the parameter 'K' in K-NN Algorithm.
3. A single variable 'M' which is the coefficient in fuzzy algorithm.
4. A trainig matrix 'P' which contains a set of feature vectors. Each vector
is in a column of the matrix.
5. A class membership matrix 'T' which contains the membership values of the
training set vectors to the classes.
6. An input matrix 'U' which contains a set of unclassified feature vectors.

The main program uses the FuzzyKNN routine to classify each one of the input
vectors and saves the results (the classes that these inputs belong to) in a
file called "foutfile.mat". This file is in Matlab format. This file contains
a single variable called fresult. It is a vector of the classes.

This program can be called from dos, or within Matlab by using dos escpae
character '!'. An example Matlab script file that shows how this program can
be used is included in the file "fknntest.m".

*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <conio.h>

#define INPUTFILE "fdatafil.mat"
#define OUTPUTFILE "foutfile.mat"

// Function Prototypes ----------------------------------------

void FuzzyKNN(double *Input, double *Samples, double *Lables, double *Result);
double FindDistance(double *vec1, double *vec2);
double Maxd(double *vec, int *index, int Length);
int FindMax(int *vector, int *count, int Length, int Max);
int loadmat(FILE * fp,int *type, char *pname, int *mrows, int *ncols,
                int *imagf, double **preal, double **pimag);
void savemat(FILE *fp, int type, char *pname, int mrows, int ncols,
```

```
                int imagf, double *preal, double *pimag);


// Global variables, these variables will be set by reading matlab file ------

int Classes;        /* the number of classes */
int features;       /* Number of features in a class */
int KK ;                                    /* K in K-nearest neighbors */
int SampleSize;                             /* Number of Labled Samples */
int TestSize;
double M;                                                   /* Coefficient in fuzzy
algorithm

//-------------------------------------------------------------


/*-------------------------------------------------------------*/

void main()
{

        double *Lables;
        double *KP;
        double *input;
        int i,j;
        FILE *fp;
        char name[20];
        int type, imagf;
        double *Samples, *isamples;    // isamples is for imaginary part of the matrix that is not used in
here.
        double *Testdata;
        double *result;          // pointer to the result matrix
        double *iresult;                        // result vector of classification of a single vector

        fp=fopen(INPUTFILE,"rb");
        if(!fp) {
                printf("cannot open the file");
                exit(-1);
        }
        // read classes from the file
        loadmat(fp, &type, name, &i, &j, &imagf, &KP, &isamples);
        if(i!=1 || j!=1) {
                printf("error: You should include classes at the beginning of the file\n");
                exit(-1);
        }
        Classes=*KP;

        // read KK from the file
        loadmat(fp, &type, name, &i, &j, &imagf, &KP, &isamples);
        if(i!=1 || j!=1) {
                printf("error: You should include K at the beginning of the file\n");
                exit(-1);
        }
        KK=*KP;
```

```
// read M from the file
loadmat(fp, &type, name, &i, &j, &imagf, &KP, &isamples);
if(i!=1 || j!=1) {
        printf("error: You should include M as the thrid parameter\n");
        exit(-1);
}
M=*KP;

// read the matrix from the datafile.
loadmat(fp, &type, name, &features, &SampleSize, &imagf, &Samples, &isamples);

// reading lables from data file
loadmat(fp, &type, name, &i, &j, &imagf, &Lables, &isamples);
if(i!=1 || j!=SampleSize) {
        printf("error: Number of labels is different from the number of samples\n");
        exit(-1);
}

// read data to be classified from the file
loadmat(fp, &type, name, &i, &TestSize, &imagf, &Testdata, &isamples);
if(i != features) {
        printf("error: Training and testing matrices should have the same size");
        exit(-1);
}

// Allocate space for result vector

result = (double *) malloc(TestSize*Classes*sizeof(double));
if(!result) {
        printf("Error: cannot allocate memory for the result Matrix");
        exit(-1);
}

for(j=0; j<TestSize; j++) {          // for each input
        input=Testdata+j*features;
        FuzzyKNN(input, Samples, Lables, iresult);
//      printf("\n Memberships:");
        for(i=0; i<Classes; i++) {
                result[j*Classes+i]=iresult[i];
                printf(" %lf ", iresult[i]);
        }
}
fclose(fp);
//  printf("\n  End of classification, Now writing the result in the file");

fp=fopen(OUTPUTFILE, "wb");
if(!fp) {
        printf("Error: Cannot write the file");
        getch();
}
savemat(fp, 0, "fresult", Classes, TestSize, 0, result, result);
fclose(fp);
```

```
}

/*-------------------------------------------------------------*/
/* This is a fuzzy K Nearest neighbor classifier routine. Input is the
vector to be classified, Samples is the matrix of classified samples,
Lables is the vector of the classes that these samples belong to.
Result is the vector of membership values of Input to each class.
*/
void FuzzyKNN(double *Input, double *Samples, double *Lables, double *Result)
{
        int i,j,n ;
        int nj, k, nk;
        double *distance;
        int *index;
        double x,y;
        double *membership;                     // pointer to membership matrix
        double nsum, dsum, temp;

        /* This section builds a fuzzy membership matrix from the lables.
        Membership of each sample to the class that it belongs to is assigned
        to 1, and the membership of it to other classes is assigned to 0 */

        membership = (double *) malloc(SampleSize*Classes*sizeof(double));
        if(!membership) {
                printf("Error: Not enough memory for membership matrix");
                exit(-1);
        }
        for(i=0; i<SampleSize*Classes; i++)
                *(membership+i)=0;                      // Initializing matrix to zero
        for(j=0; j<SampleSize; j++) {
                i=*(Lables+j);
                *(membership+i*SampleSize+j)=1;
        }

        distance = (double *) malloc(KK*sizeof(double));    // allocate space for the vector
        if(!distance) {
                printf("Error: Not enough memory for distance vector");
                exit(-1);
        }

        index = (int *) malloc(KK*sizeof(int));
        if(!index) {
                printf("Error: Not enough memory for index vector");
                exit(-1);
        }

        for(i=0; i<KK; i++) {  // This loop initializes K nearest neighbors to the first K Samples
                index[i]=i;
                distance[i]=FindDistance(Input, &Samples[i*features]);
        }
        for(i=KK; i<SampleSize; i++) {    // This is the loop that finds the K nearest Neighbors
                x=Maxd(distance, &j, KK);
                y=FindDistance(Input, &Samples[i*features]);
                if(y < x) {  // This sample is closest to the input than the farthest K Neighbors
```

```
                        distance[j]=y;
                        index[j]=i;
                }
        }
        for(j=0; j<Classes; j++) {
                nsum=dsum=0;
                for(n=0; n<KK; n++) {
                        i=index[n];
                        temp=FindDistance(Input, &Samples[i*features]);
                        if(temp < 1e-10) {                              //If distance is
zero
                                Result[j]=membership[j*SampleSize+i];
                                break;
                        }
                        if(M == 2)
                                temp=1/temp;
                        else if(M != 1)
                                temp=pow(1/temp, 1/(M-1));
                        else
                                temp=0;
                        nsum += membership[j*SampleSize+i]*temp;
                        dsum += temp;
                }
                if(dsum !=0)
                        Result[j]=nsum / dsum;
        }
        free(membership);
        free(distance);
        free(index);
}


/*-----------------------------------------------------------------*/
/* This function returns the Euclidian distance between two vectors */

double FindDistance(double *vec1, double *vec2)
{

        int k;
        double distance;

        distance = 0;
        for(k=0; k<features; k++) {
                        distance += (vec1[k]-vec2[k])*(vec1[k]-vec2[k]);
//                      distance += pow(vec1[k]-vec2[k] , 2);
        }
        return distance;
}

/*-----------------------------------------------------------------*/
/* This function finds the biggest element of an array.  It returns that
value and also returns the index to that element in index.
*/
```

```
double Maxd(double *vec, int *index, int Length)
{
        int i,j=0;

        j=0;
        for(i=1; i<Length; i++)
                if(vec[i]>vec[j]) j=i;
        *index=j;
        return(vec[j]);
}

/*-  -  ----------------------------------------------------------*/
/* Thi. function finds a number that is most often repeated in an array of
integer values, and returns that number. Length of array shoud be less than
100.  It is supposed that number is an integer greater than zero.
vector is a pointer to the array.  count is the number of times that the
number is repeated.  Length is the length of the vector.
*/

int FindMax(int *vector, int *count, int Length, int Max)
{

        int i, j, m;
        int t[101];

        if(Max>100) Max=100;
        for(i=0; i<Max+1; i++)
                t[i]=0;
        for(i=0; i<Length; i++)
                t[vector[i]]++;
        m=t[1];
        j=1;
        for(i=1; i<Max+1; i++) {
                if(t[i]>m) {
                        m=t[i];
                        j=i;
                }
        }
        *count=m;
        return (j);
}
```